Journal of the ISTE Special Interest Group for Logo-Using Educators

LOGO EXCHANGE

December 1989 Volume 8 Number 4

Snowmen

Children make snowmen with snowflakes in winter time. Children think making a snowman is fun.

International Society for Technology in Education

ISTE Publications

Our new formula is no secret.

Introducing *Formula Vision*, a welcome sight for teaching math and science, grades 4-12. *Formula Vision* is a one-screen, 15-cell spreadsheet program capable of holding even advanced secondary math problems. Use *Formula Vision* with a large monitor–the 40-column text allows the whole class to participate.

Formula Vision lets you name the cells and formulas. "Area=Height x Width" appears as "Area=Height x Width" instead of the often confusing "+C4 * C5." Units in each cell can be labeled and different units of measure converted as well.

Envision your students involved in problem solving with *Formula Vision*.

Formula Vision runs on the Apple II with 64K memory. MS-DOS version available soon.

Classroom site license: $29.95
School site license: $49.95

ISTE, University of Oregon, 1787 Agate St., Eugene, OR 97403-9905; ph. 503/686-4414

They have creativity in their little fingers.

Probably more than you realize. To help you capture some of that creativity, Randy Boone has compiled *Teaching Process Writing with Computers*.

This collection includes:

- research and position papers
- lesson ideas
- keyboarding
- reviews
- articles from *The Computing Teacher* and *The Writing Notebook*.

Randy Boone highlights ideas for teaching process writing with computers in the introduction, making *Teaching Process Writing Computers* an ideal tool for university courses, inservice workshops, or as an effective resource for classroom teachers. **$15.95**

Make creativity a state of being.

ISTE, University of Oregon, 1787 Agate St., Eugene, OR 97403-9905; ph. 503/686-4414

LOGO EXCHANGE

Volume 8 Number 4 Journal of the ISTE Special Interest Group for Logo-Using Educators December 1989

Publisher
International Society for Technology in Education

Logo Exchange is the journal of the International Society for Technology in Education Special Interest Group for Logo-using Educators (SIGLogo), published monthly September through May by ISTE, University of Oregon, 1787 Agate Street, Eugene, OR 97403-9905, USA.

POSTMASTER: Send address changes to Logo Exchange, UofO, 1787 Agate St., Eugene, OR 97403. Second-class postage paid at Eugene OR. USPS #000-554.

Contents

ISTE Basic Membership
(8 issues of *Update* and *The Computing Teacher*)

	U.S.	Non-U.S.
1 year membership	28.50	36.00

SIGLogo Membership (includes *The Logo Exchange*)

	U.S.	Non-U.S.
ISTE Member Price	25.00	30.00
Non-ISTE Member Price	30.00	35.00

Send membership dues to ISTE. Add $2.50 for processing if payment does not accompany your dues. VISA and Mastercard accepted. Add $18.00 for airmail shipping.

From the Editor

Happy Holidays!

You awaken very early on a crisp, clear holiday morning. You feel refreshed and renewed. The hectic pre-vacation days at school are over and now it is time for family and friends. You wander into the living room of the silent house. There in the corner of the room is a pile of gifts that you have never seen before. Curiously, you go over to look at them. To your amazement, they are all addressed to you! None of the tags say who they are from. Who could have left them? Their colorful paper and sparkley bows are a temptation too hard to resist. . Should you wait until the family arrives, or should you take a peak?

Like a naughty child, you tentatively shake the colorful green and white package on top. Something in it rattles just a bit. It 's rather heavy – better be careful. Perhaps it is breakable. Curiosity overcomes you. You carefully remove the wrapping paper and open the box. What is this? A new version of Logo that works on the computers in your classroom. . A quick look at the manuals gives you an overview of the features. Wow! This is the Logo you have always dreamed of. It has amazing capabilities which you and your students will love using. But wait. You haven't heard any announcements about new Logo versions. Where could it have come from? Who could have given this to you? Well, in any case, later today you'll simply have to try it out. How much easier it will make your classroom use of Logo. How much more your students will be able to accomplish!

Now your curiosity is really peaked. What could be in these other mysterious boxes? Still no sounds from the family, so you decide to be even more daring. The next box, a flat one wrapped in blue paper with silver ribbons, tempts you. Again, you tentatively open it. It's a brochure for a new computer system – one you had no idea was available. Look at the features! It will even run that new version of Logo that you just opened. And there's so much memory. And fantastic sound capabilities. And look at the colors available. If only you had machines like that at school. But what's this other piece of paper? Hmmm. It's a letter to all teachers in your school. All of the old equipment is being replaced with these new models. Every classroom will have a color laser printer, a modem, and a network of its own. It's a dream come true. There will even be a large library of software provided. Someone was really thinking ahead. Just imagine how much more your students can learn. You simply can't believe it! What a wonderful holiday gift. How did your school manage it?

Now you know you can't stop. The next box, a big one with red paper and gold bows is almost cubical in shape. This time you rip off the paper and ribbons – after all, these packages are clearly for you. What is this? A doll house? You look closer. No, not a doll house – more like a doll school. No, it's your classroom, except that it has been remodeled. Look at all of the space, the storage cabinets, the facilities. What a neat science corner, and there's a well designed space for those new computers, and every kind of audio visual aide imaginable. There's even a computer station built right into your desk. Then you notice the memo – the school is being remodeled over the holidays and you will return to a classroom that looks just like this. Wow! Just think how much more productive you will be able to be with this wonderful new facility. You won't have to waste your time using makeshift equipment or shuffling materials to find more storage space. Your students will learn so much more.

There's just one more of these mysterious boxes. It's a strange one – sort of shimmery in it's gold paper. Almost hard to look at. Wait. This one has a tag. It says "Open with care. Contains perfect students." "Perfect students?" What is a perfect student? The difficult moments you have had with various members of your class flash through your mind. Their faces are clear in your mind's eye. "Perfect students?" Hmmm. You open the box – and somehow you see your class, all perfectly behaved. Time flashes past – the school day flows without a single moment of difficulty in that shiney new classroom with the beautiful new equipment. As the children leave the room, you feel empty. Did you really teach these perfect students anything? Did it even matter that you were there? Wait a minute. How long have you been sitting here in the midst of boxes and wrapping paper. You mutter "perfect students...perfect students...perfect students..."

...and you open your eyes. It's morning. "Perfect students." Wait. You rush to the living room. There are no mysterious packages. "Perfect students." You sit down in a nearby chair, feeling dejected. It was such a wonderful dream. Everything you could possibly want to make education in your classroom better. All those things that you and your colleagues complain about were fixed. If only the dream had been true. You'd be so much more effective.

Perfect students, a perfect classroom – a dream come true. But wait a minute. Those perfect students weren't particularly interesting. Did they even need you? Did they learn anything as they sat angelically in that high tech classroom?

You're a teacher – a teacher of real kids. You don't need all of those wonderful facilities to do your job. Certainly there's more to education than magnificent hardware and software. You have a great deal to give your students that no technology or facility can ever give. You encourage, you nurture, you motivate, you criticize, and you love.–whether they are perfect or not. No computer, no piece of software, no physical plant can do that. It's up to you and your colleagues to make education better. You mood lifts. You resolve to return to the new year refreshed and renewed, knowing that your role is central if education in your school, city, or country is to improve.

Sharon Yoder
ISTE
1787 Agate Street
Eugene, OR 97403
CIS: 73007,1645 BITNET: Yoder@Oregon

About the Cover

This month's cover was created by Christina Coffin and Tara Engan. These first graders were students in a LogoWriter lab in Colts Neck Township Schools. Their teacher is Jandy Bird, whose article on cooperative learning using Logo is found in this issue.

Monthly Musing

ST, HT
by Tom Lough

"There's a good chance we'll see some turtles tonight!" Kevin McIntyre shouted above the roar of the twin Evinrude outboards as the boat neared Buck Island. Kevin was a naturalist working on temporary assignment with the National Park Service in the US Virgin Islands. His project was to observe and record any nestings of sea turtles on the rocky beaches of the Buck Island preserve.

I had taken the evening off from the Logo in Paradise course I was teaching with Glen Bull and Dennis Harper at the University of the Virgin Islands. Dennis' son, J, and I had been lucky enough to be invited to Buck Island as VIP's. We found out that VIP meant Volunteer In the Park, and it qualified us to carry equipment, hold flashlights, and perform other acts to assist Kevin. Just then, being a VIP meant helping to moor the boat and prepare for the nocturnal beach patrol.

Our first job was to wait for dark. We clambered over the rocks, trees and sand to the easternmost point of the nesting beach, spread out plastic sheets and sat down on them. While the scarlet sun sank into the ocean, I began thinking about what might happen later. Somewhere in the water nearby there may be a female sea turtle waiting for just the right moment to clamber ashore, dig a hole and lay her eggs. I couldn't wait!

Of course, it is no surprise that I was also thinking abut Logo! All of a sudden, the commands SHOWTURTLE and HIDETURTLE took on new meaning for me. I thought about a set of procedures which seemed appropriate for the situation.

```
TO LAND?
IF XCOR > 0 [OUTPUT "TRUE]
OUTPUT "FALSE
END

TO OCEAN?
IF XCOR < 1 [OUTPUT "TRUE]
OUTPUT "FALSE
END

TO MOVE :DISTANCE
REPEAT :DISTANCE [FORWARD 1 CHECK]
END

TO CHECK
IF LAND? [SHOWTURTLE PENDOWN]
IF OCEAN? [HIDETURTLE PENUP]
END
```

Use MOVE instead of FORWARD. When the turtle is on "land," then you can see her tracks. When she is in the "ocean," you can't really tell exactly where she is! What activities can you create using these procedures?

Meanwhile, the sky had grown dark and was speckled with stars. Kevin quietly announced that our trek was about to begin. We trudged silently up and down the beach for what seemed like an eternity. Suddenly, in the wee small hours, Kevin froze! Motionless, he concentrated on the sounds and smells of the night. Then he signalled for us to sit down. A turtle was nesting nearby! This was definitely SHOWTURTLE time!

After about fifteen minutes, Kevin made a brief foray into the trees next to the beach and returned to whisper excitedly, "She's a hawksbill! She's digging her hole now and will be laying the eggs in about 30 minutes!" While we waited on the beach, I was overwhelmed by the idea that these magnificent reptiles had been performing this same act for millions of years. I felt very privileged for the observation I was about to make. Soon Kevin announced that we could approach the turtle. She had settled into a hole as big as her shell (about 80 centimeters across) and then had dug a smaller deeper hole with her hind flippers for the eggs. When sea turtles are in the act of laying their eggs, they go into a sort of trance, so we could get quite close without disturbing her. By the light of Kevin's flashlight, we could see the glistening spheres of her eggs as they dropped softly into the nest, 160 of them in all.

Hesitantly, I reached out my hand and touched her shell, then a flipper. It was a magical moment. All too soon, Kevin motioned for us to return to the beach. The turtle began to cover her precious eggs with sand. After about 30 minutes, her job was finished and she clambered down the embankment toward the water. I couldn't help but follow. She showed no real fear of me. As she reached the water, she paused as if to bid me farewell. She entered the surf, slowly at first, then with increasing speed. In a sudden swirl of froth, she was gone. HIDETURTLE.

I have thought about that night many times. Thanks, Kevin, for that close encounter with a turtle of the real kind. We will be impoverished indeed if we allow these wonderful creatures to become extinct.

ST FD 100!

Tom Lough, Founding Editor
Box 394, Simsbury, CT 06070

PS: If you or your students wish to become active in helping to save endangered turtle species, write for information to: Help Endangered Animals, Ridley Turtles, PO Box 681231, Houston, TX 77268.

Logo Ideas

Time, Numbers, and Other Things
by Eadie Adamson

How Big Can It Be?

Last spring during one of my fourth grade classes, we began talking about numbers, specifically, how large a number could LogoWriter handle? It was a good chance for my grade to think about how to solve a problem. It didn't take them long to choose a strategy.

"Let's just try a really big number," said one. "If Logo doesn't complain, we'll make it bigger. If it does complain we'll make it smaller, till we find the largest number we can use without an error message."

Off they went, using the **print** command. That task did not take long (I leave it to you and/or your classes to find the number!)

"That's great." I said. "I wonder how long it would take Logo to count to that number? How could we find out?"

Counting Up!!

Our first strategy was to write a procedure like this:

```
to countup :num
print :num
countup :num + 1
end
```

We began with **countup 1**. Much to our surprise, before the class time was up, we got an error message: **No more room for text on the page!** I helped my students modify the procedure so that it cleared the text each time, by adding the command **ct** to the **countup** procedure just before the recursive call. The procedure then looked like this:

```
to countup :num
print :num
ct
countup :num + 1
end
```

Our adventure went on for several weeks, with the computer working weekends too. Students began coming in at all times of the day to see how our experiment was developing. After several weeks, with nearly everyone cooperating about leaving the test computer empty, sadly, one evening after school someone turned the computer off. We didn't discover it until the next morning. By that time we gave up on the experiment. Not enough time was left in the school year to start again. Our question was never answered, but we had an idea that it took a really long time. We also found that the computer, though it was taking a long time, could count much faster than we could!

A Similar Question and a Request for Help

Recently I received a letter from Robert Macdonald (see November *LX* for an article about some of his work with students). To my delight, he had been working on a similar experiment with his students, but in his usual inventive way had combined it with other number investigations. Here's what his students wrote about their problem:

> One of the first problems we encounter in our fourth grade mathematics class is: How much is a million? Earlier, during the third grade we attempted to collect one million bottle caps. It was quite a job.
>
> Mr. Macdonald suggested we find out how long it would take a computer to count to a million.

The students proceeded to tell about their adventures running a race with three computers and a BASIC (!!) program which took between three and a half to four hours to complete the counting...

> However, we like to use LogoWriter in our class not BASIC. When we programmed our initial program we were lucky to get up to 300 in a crawl:

```
to countup :input
if :input = 100001 [stop]
insert :input insert char 32
countup :input + 1
end
```

The class contacted Sharon Yoder who suggested periodically clearing the screen, giving the students another procedure to try:

```
to count.and.clear :input :lines
if :input = 1000001 [stop]
if :lines = 18 [make "lines 0 ct]
print :input
count.and.clear :input + 1 :lines + 1
end
```

Sharon's solution sets up a counter. After every eighteen lines, the text is cleared. Meanwhile, besides counting, the procedure is checking for a maximum input. At each call to the procedure, the number (**:input**) and the number of lines is

Logo Ideas--continued

incremented by one. The two stop rules operate each time, one checking whether the maximum number has been exceeded (and, if so, stopping) and the other keeping track of the number of lines, clearing the text when the number of lines equals eighteen.

Robert's students said "It took our computer 47 hours and 3 minutes to finish the count."

The class is seeking help to find a speedier way for Logo to count to a million. I had several other suggestions for them. First, write the counting procedure like this:

```
to countup :num
if :num > 1000000 [stop]
print :num
countup :num + 1
end
```

This procedure, running on an Apple II GS at its FAST speed, will stop in approximately 15 hours. That's faster, but it won't beat the students' BASIC record. I had one other thought, however. Each time a procedure asks Logo to check something or to do something, it obviously takes a small amount of time. All we really want to know, if I understand their question, is: how fast can Logo count to a million? The issue is: what takes time and what do we need to do? Do we really need to print the answer? If not, perhaps the solution is as follows (and takes about 30 minutes on a GS):

```
to countup :num
if :num > 1000000 [print :num -1
   stop]
countup :num + 1
end
```

Then you can program a control key that will show you the number, so that there is a way to check on the progress:

```
when "z [show :num]
```

I consulted with Michael Tempel, Director of Educational Services for Logo Computer Systems, to see what he might add. Michael suggested that comparing the speed of programming languages was not really the point, although the time it takes to count to a million is obviously part of what interested the children. BASIC, Michael pointed out, is a much less complex language than LogoWriter. For real speed, he suggested writing the program in assembly language. Comparing two languages to another is like comparing apples and oranges. What Michael found much more interesting and puzzling was the wide discrepancy in speed in the two Apple IIe computers the students used to test the Basic program: one took 3 hours and 25 minutes, while the other took 3 hours, 52 minutes and thirty seconds — a difference of nearly thirty minutes, much more than would seem logical.

Michael had another idea for solving the problem. Working in the Command Center, type the following:

```
make "num 0
repeat 999 [repeat 999 [make "num
   :num + 1] print :num]
```

This should cause the computer to print the current number at every thousandth time it increments the number. The counting should stop at 1000000. Will it work? Try it out before reading on!

I adjusted Michael's idea a bit to get the result I wanted:

```
make "num 0
repeat 999 [repeat 1000 [make "num
   :num + 1] print :num]
```

What's interesting about a problem like this is that it gets us thinking about the most efficient way to write a program to solve a problem. This is an idea students don't often appropriate easily. Usually something works and they will see little reason to make their procedure more elegant or more direct. Here is an instance in which going right to the simplest way to express the problem becomes important, since we are interested in speed. We then need to look more carefully at what we are doing, eliminate the unnecessary steps, tighten things up, perhaps even try a different approach. The experiment creates a nice opportunity to talk about programming style as well. Bob Macdonald, Michael, Sharon, and I each had a slightly different idea about how to approach the problem. It's an interesting lesson for students: ask for help when a problem seems insoluble. In this case several adults attempt to solve the problem by contributing a variety of solutions. I can't think of a better way to make the point that there's no one single correct way to do something in Logo, although one solution may end up being more efficient than others (the efficiency depending upon what the criteria is for a satisfactory solution). It's also good for students to see adults struggle to solve a programming problem, too!

Now...how long would it take LogoWriter to count to a number it couldn't handle? Can we express the problem better? I think we'll try again!

Other inventive solutions or suggestions are welcome. Write to Robert Macdonald's fourth grade class at:

Mr. Macdonald's Fourth Grade Room
Meridian Elementary School
26700 Meridian Road
Grosse Ile, Michigan 48138

The children would love to hear from you. (P.S. Please send me a copy of any creative solutions, too!)

On to Chaos

I had planned to write a column on the subject of chaos, but Phil Firstenbaum got there first (September *LX*. "Order from Chaos"). I was interested in his solution to this problem since mine was somewhat different (see below). This summer at the Eurologo Conference at the University of Ghent, Ghent, Belgium, several other people talked about experimenting with the chaos game with their classes. The idea of playing the game with Logo is clearly in the air everywhere! I thought, as a follow-up to Phil's article, I might share a few more ideas on the subject.

I first played with this idea when Michael Tempel and I flew together from New York to the West Coast Logo Conference in Los Angeles. Michael had a portable computer with him and a long plane ride seemed an ideal time to experiment. Later I too saw the Nova show on chaos and realized how much the audience missed, since they had no opportunity to see the intervening steps, to see a really graphic version of order finally emerging from a chaotic mess of dots. Here's the program I wrote (notice I added a control key so that I could print the screen as the process as it developed):

```
to chaos
cg
when "Z [printscreen]
put.spots [1 2 3]
doit
end

to put.spots :turtles
if empty? :turtles [stop]
tell first :turtles
pu setpos list random 280 random 160
dot
put.spots butfirst :turtles
tell 0
pu
end

to doit
name 1 + random 3 "turtle
seth towards ask :turtle [pos]
forward (distance
   ask :turtle [pos]) /2
dot
doit
end

to dot
pd
forward 0
pu
end
```

* * The procedures distance and square are used to get the distance of the main turtle from a given set of coordinates: * *

```
to distance :away
output sqrt (( square (xcor - first
   :away)) +
     (square (ycor - last :away) ))
end

to square :num
output :num * :num
end
```

European Connections

At the Eurologo conference one presenter shared his investigations of chaos. Several of us talked about experiences and ideas for exploring the chaos game with our students. For instance, are there any patterns other than triangles that act this way? The program could be adjusted to take a predetermined list of points to start with, instead of picking random spots, as Michael and I did. The turtle could then turn randomly to one of the points from the longer list, move halfway and deposit a dot. One could thus plot out an infinite variety of shapes and run tests to find the answer. This would be fun to do as an overnight run on several computers, each testing a different configuration of dots. Another idea to investigate, which won't take long, is to find out what happens if the pen is left down, rather than lifted until it makes a dot. Again, are there configurations which do give you interesting patterns with this method?

Curioser and Curioser

Curiously, at the West Coast Logo Conference last February, Paul Goldenberg showed a fractal design which produces almost the same result as the completed chaos game picture. Here's a procedure which generates a similar fractal:

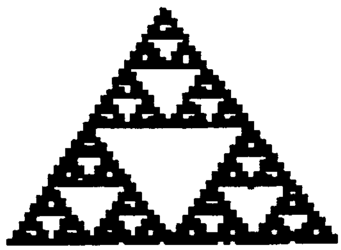
Logo Ideas – continued

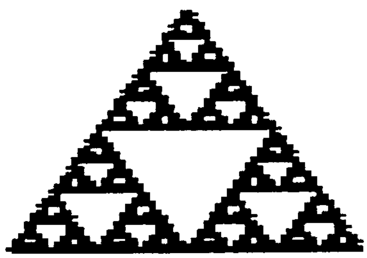

```
to tri :comp :side
if :comp = 0 [stop]
repeat 3  [forward :side/2 left 120
   (tri :comp - 1 :side /2 ) right
   120 forward :side/2 right 120]
end
```

I used:

```
pu
back 50
pd
left 30
tri 6 60
```

to generate my fractal drawing. (Can you tell which is the fractal drawing and which is the drawing from the chaos experiment?)





(The top drawing is from the chaos experiment, the bottom one from the **tri** procedure.)

Eadie Adamson
Allen Stevenson School, 132 East 78th Street
New York, New York 10021

A Primary Instant Program Leads to Theme Booklets

by Glenda Bentz

Typing can be a problem for primary-aged children (K-2). As they are learning about the turtle, they should need to use minimal typing skills. Kindergarten and first grade age children in my school use the one stroke instant program with a slow turtle that comes with LogoWriter. When second graders come to the computer lab, they use my own primary form of instant Logo which prints the "real" commands in the Command Center. These commands are the same ones they used in Kindergarten as F, B, R, L—with a slow turtle—, but now they see the numbers and real commands. They do not have to hunt and peck to draw. Instead they use the up and down arrows to place the cursor on a command and then press the return key. Here are the commands they see in the Command Center when my instant program is run.

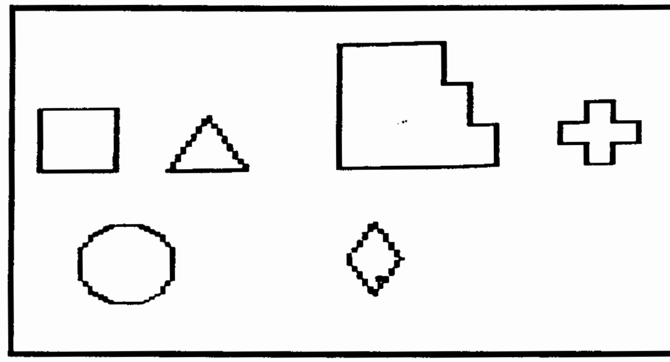
```
FORWARD 10
BACKWARD 10
RIGHT 30
LEFT 30
```

The FORWARD 10 command immediately scrolls up beyond view. However, the children quickly learn to hold down the up arrow to bring the FORWARD command back into view. They will often need this skill in the future. However, if the last CHAR 13 is left out of the procedure called 'C' shown below, the cursor will wait on LEFT 30 and leave all four commands in view. You might want to try the program both ways, before you decide which you prefer.

Getting Started

During their first computer classes, the second graders use this program with a startup which forces them to begin a new drawing each lab period. If students need a fresh list of these basic commands, they go to a blank line in the Command Center by using the down arrow key and type 'C' followed by the Return key. If they want to start over with a clean screen and a fresh set of commands, they type 'D' and Return on any blank line in the Command Center. The C and D keys, arrows, and Return are the only keys they need to draw with the turtle.

I provide ideas for figures they might draw—particularly polygons and other regular shapes like fat crosses or stairs. Of course, they may always draw anything they would like to draw. If their learning is not progressing (i.e., diagonal lines and no noticeable shapes), I may add a challenge such as asking them to make as many polygons on the screen as they can. The big challenge for them is learning to draw a circle with these commands. FORWARD 10 AND RIGHT 30 repeated alternately will make a small circle. When the students ask how to draw a circle, I ask them to walk in a small circle. What are they doing? The usual answer is "walking a little" and "turning a little". Now they must put "walking a little" and "turning a little" into turtle commands.



Saving Student Work

During the last several lab classes, I make a directory,

```
CREATEDIR "ANIMALS
```

remove the startup procedure from the instant program, and save the new program by the teacher's name for each student in each class at each computer station. In the classroom, the teacher could make a file for each student by names or initials (no directory is needed in the classroom). Remove the startup procedure from the instant program and use the NAMEPAGE command to name a student's page and Escape to save. Continue to rename and save the pages until all the students are assigned their own page. Now each student can press Escape and save his work for another session for the first time.

Making a Booklet

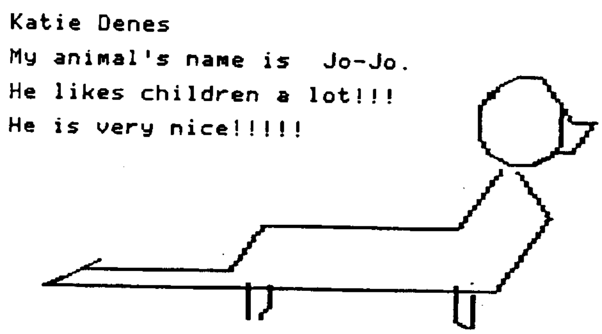
The final assignment is to draw an imaginary animal and then write about their original animals on the screen. When the assignment is made, we learn the Turtle Move keys (Open Apple-9) to allow moving the turtle without drawing, using the arrow keys. By now I can ask the students when making the assignment, if they can think of anything which will help them to draw more easily. They know they need a way to move the turtle without drawing and so we learn what we need as we progress.

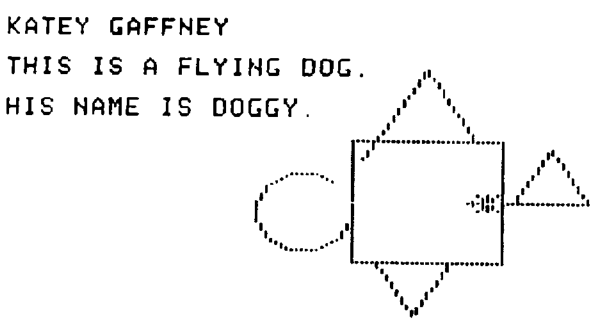
At the end of each session, students need to be reminded to press Escape to return to the Contents page to have their work saved. From Kindergarten on, each student is expected to write his name at the top of the screen before beginning to draw with the turtle. This transition to the word processing stage is almost automatic, and many young students ask to write stories in their "original" spelling forms long before we

Benz –continued

draw and write our class booklet of imaginary animals. If your students are not using the Up and Down command (Open-Apple U and Open-Apple D) to allow them to write on the screen, you will need to teach these commands as well.

When the second grade imaginary animals are all in print form, I use the Logo Alphabet from ISTEto title the book, paste it up, and copy it on legal paper using both sides of a page. The book is shared in the classroom, so each student can read and appreciate each and every animal. Sometimes the classroom teacher uses the books for a grammar lesson by asking students to find misused words such as 'to' for 'two' or to find misspelled words.





Two Imaginary Animals

Here is my Logowriter Primary Instant program listing without STARTUP:

```
TO BEGIN
SET.UP
DIRECTIONS
END

TO SET.UP
HT
CT
RG
CC
END

TO DIRECTIONS
PRINT [USE ARROW KEYS TO PLACE]
PRINT [BLINKING BOX ON COMMAND.]
PRINT [PRESS RETURN TO MAKE TURTLE]
PRINT [DO THE ACTION.]
PRINT []
PRINT [P = PRINTSCREEN]
PRINT [H = HELP PAGE]
PRINT []
PRINT [PRESS RETURN FOR COMMAND]
PRINT []
PRINT [PRESS RETURN NOW TO START]
PRINT READCHAR
SET.UP
ST
END

TO D
ST
RG
CC
RECYCLE
C
END

TO P
PRINTSCREEN
END

TO H
BEGIN
END

TO C
CC
(TYPE [REPEAT 10 [FORWARD 1]]
   CHAR 13)
(TYPE [REPEAT 10 [BACKWARD 1]]
   CHAR 13)
(TYPE [REPEAT 30 [RIGHT 1]] CHAR 13)
(TYPE [REPEAT 30 [LEFT 1]] CHAR 13)
END
```

Note: Leaving out the last CHAR 13, will leave all the commands in view in the Command Center.

Glenda Bentz, Elementary Computer Coordinator
Oak Hill School 801 South Spoede Road
St. Louis, Mo 63131
School Phone: (314) 432-2021 ext: 257
Home Phone: (314) 961-3015
Home Address: #2 Villawood St. Louis, Mo 63119

Beginner's Corner

Holiday Wrappings!
by Dorothy Fitch

Why not use Logo to help you with your gift-wrapping! This month, we'll design some wrapping paper and gift tags that are fun to make and useful, too!

Good things come in small packages, they say, which are about all this wrapping paper will cover! Of course, you can tape several printed sheets together to make larger sheets.

If your students are like most, they probably like to spend inordinate amounts of time "wrapping" the turtle around the screen. They turn the turtle at some angle and move it forward thousands and thousands of turtle steps. Why not turn that "wrapping" into wrapping paper? This may satisfy their apparent need for this activity and provide them with a useful end-product as well.

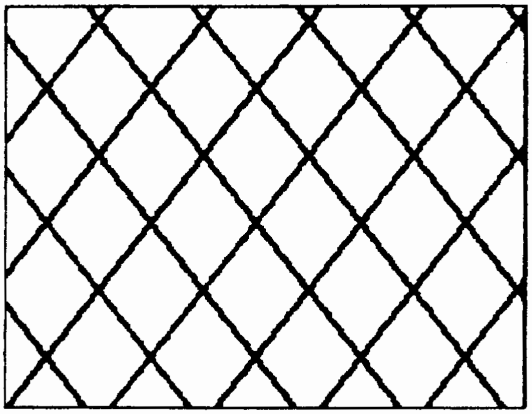
The following procedure is quite simple, yet it produces intriguing results.

```
TO WRAPPER :NUMBER      <-this procedure
                        takes a number
                        (angle) as in put
DRAW                    <-erases the screen
SETHEADING :NUMBER      <-points the turtle
                        at the specified
                        angle
FORWARD 10000           <-turtle wraps
                        around the screen at
                        that angle
SETHEADING -:NUMBER     <-turtle points to
                        the negative of the
                        angle number
FORWARD 1000            <-turtle wraps in
                        the new direction
END
```

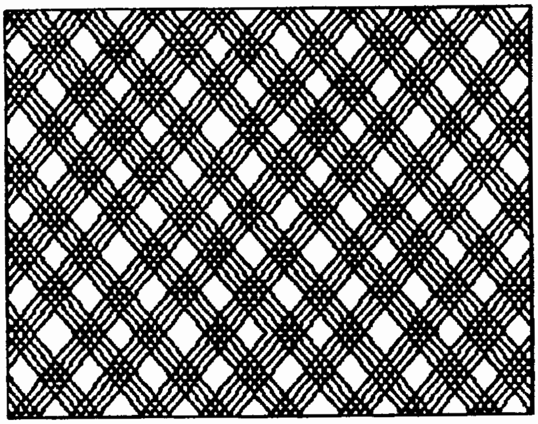
For example, if you type WRAPPER 35, the turtle will go forward 10000 steps at an angle of 35°. Then it will go forward at an angle of -35°, which is the same as 325°. (Remember that the SETHEADING command doesn't rotate the turtle to the right or left by some number of degrees; it points the turtle at a specific fixed angle.)

Some Things to Try:

- Experiment with different input angles. Numbers between 40 and 50 give particularly pleasant designs.
- How do inputs of 45, 135, 225 and 315 relate to each other?
- Why do numbers that are close together often give strikingly different (and unexpected) results? For example, these following figures (1 and 2) are the results of WRAPPER 35 and WRAPPER 36.
- Add color by inserting the line `PENCOLOR 1 + RANDOM 5` before each FORWARD instruction.



Wrapper 35



Wrapper 36

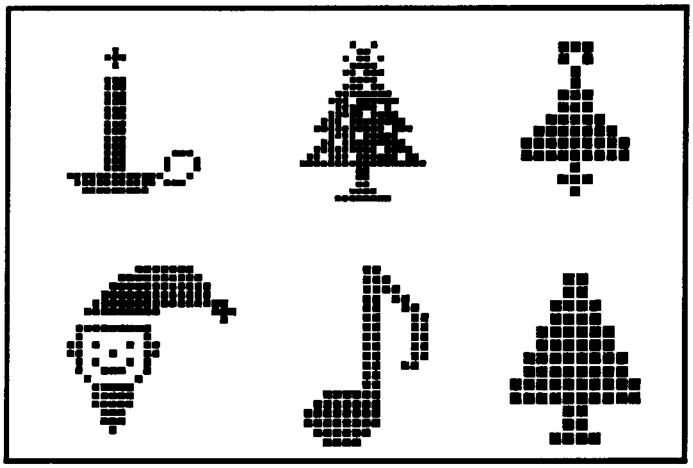
The next set of programs and screen prints were created using Terrapin's Logo PLUS, but other versions of Logo that support shapes will work as well.

Even if your version of Logo doesn't have a shape editor, you can create small designs that serve the same purpose. The design will just take longer to be drawn on the screen! Be sure

Beginner's Corner--continued

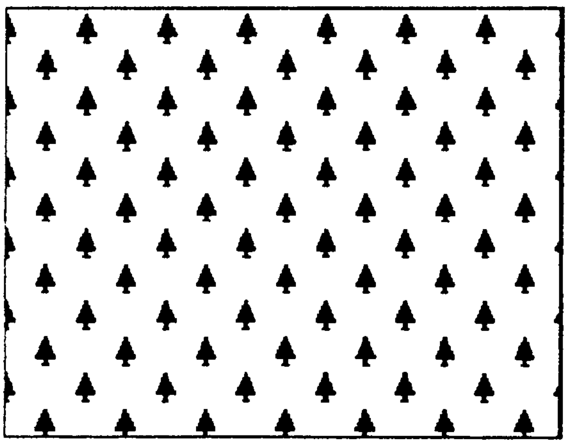
that the designs you draw to replace the shapes are "state-transparent." That is, the turtle must begin and end the design in the exact same spot and pointing in the same direction.

First create some shapes using your shape editor. Here are some designs to get you started.



Producing a neat array of shapes is simpler than you might think. The following procedure stamps shapes in a pattern of diagonal lines, as in figure 3.

```
TO PAPER
PENUP
HIDETURTLE
REPEAT 84 [STAMP RIGHT 90 FORWARD 20
   LEFT 90 FORWARD 20]
END
```



Before you run the PAPER procedure, use SETSHAPE to set the turtle's shape to one of your new designs. For example, if the shape you want to use is number 4, type SETSHAPE 4. Your turtle will assume the new shape. Then type PAPER to draw the wrapping paper design. You may want to move the turtle slightly before you run the PAPER procedure so that your shapes don't wrap around the edge of the screen. To do this, simply type PENUP and move the shape with FORWARD, BACK, LEFT and RIGHT commands until it is where you want it.

Remember that stamping a shape on top of itself makes both shapes disappear, so if you type SHOWTURTLE when your pattern is finished, the central shape will vanish. Hide the active turtle to make it reappear. When you print the design, all your shapes will be there! (Use PRINTSCREEN 1 1 for a small-sized screen print; use PRINTSCREEN 1 2 for a double-sized screen print. (For a color printout, use the SMALLCOLOR and LARGECOLOR utilities programs.)

If you don't have a shape editor and have instead created a Logo procedure that draws a small design, modify the PAPER procedure as follows:

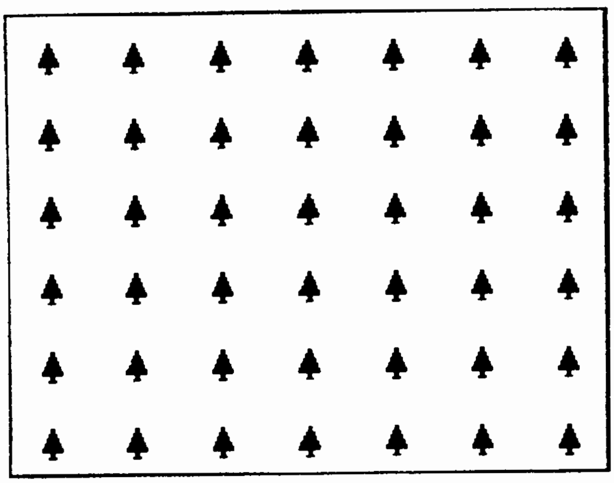
```
TO PAPER
HIDETURTLE
REPEAT 84 [STAR PENUP RIGHT 90
FORWARD 20 LEFT 90 FORWARD 20
PENDOWN]
END

TO STAR
REPEAT 8 [FORWARD 20 RIGHT 135]
END
```

If you want a border around the edge of your screen, try a procedure like this:

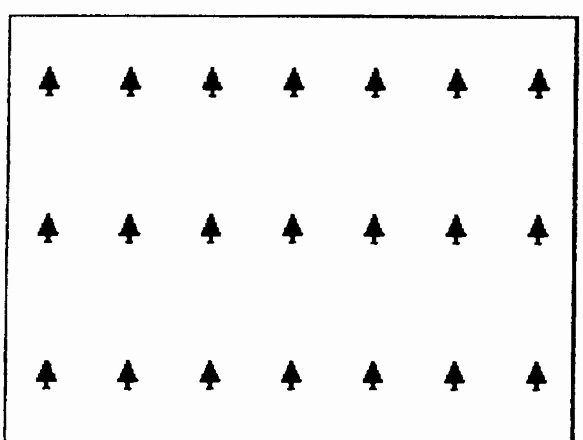
```
TO BORDER
PENUP
SETXY -140 120
PENDOWN
SETXY 139 120
SETXY 139 (-119)
SETXY -140 (-119)
SETXY -140 -120
END
```

Other variations on the PAPER theme cause slightly different results. Since the graphics screen is designed around the magic number 20, you can experiment moving the turtle 20 steps (or multiples of 20) between stamps. This will bring the turtle right back to where it started from and you'll have a symmetrical, organized pattern.

```
TO PAPER2
PENUP
HIDETURTLE
REPEAT 42 [STAMP RIGHT 90 FORWARD 40
   LEFT 90 FORWARD 40]
END
```

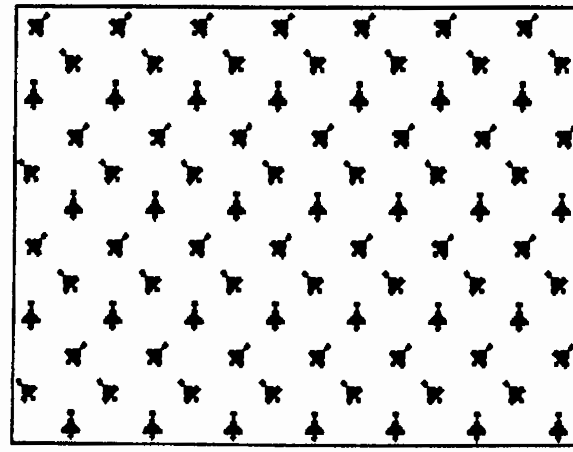


```
TO PAPER3
PENUP
HIDETURTLE
REPEAT 21 [STAMP RIGHT 90 FORWARD 40
   LEFT 90 FORWARD 80]
END
```



As you can see, there are many patterns and designs you can make by making very small changes to the program.

Challenge: try varying the position of the shape before you stamp it, as in the swinging bell pattern below.



```
TO BELLPAPER
PENUP
HIDETURTLE
REPEAT 28 [RIGHTBELL
   MOVE STAMP MOVE LEFTBELL MOVE]
END

TO RIGHTBELL
RIGHT 45
STAMP
LEFT 45
END

TO MOVE
RIGHT 90
FORWARD 20
LEFT 90
FORWARD 20
END

TO LEFTBELL
LEFT 45
STAMP
RIGHT 45
END
```

Again, you may want to move the turtle before you give the BELLPAPER command, and add a border to your finished design.

Now for some Tags!

It is also easy to create gift tags using a combination of turtle graphics, stamped shapes and different text fonts on the graphics screen.

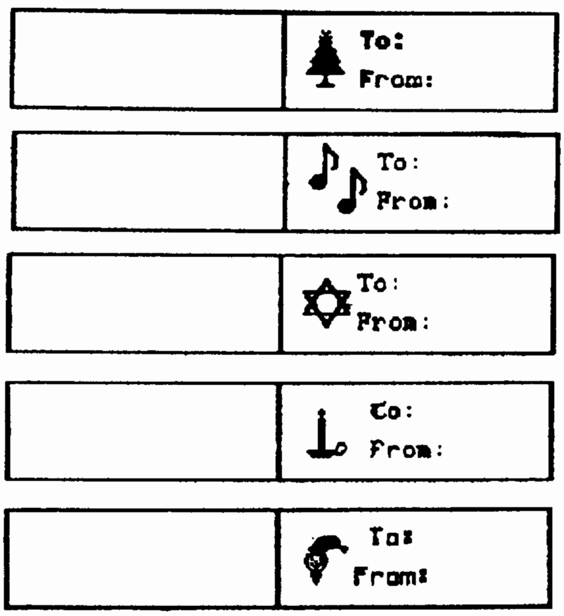
Beginner's Corner –continued

This procedure draws the outline of a tag. After you've printed your finished tag, just cut along these lines, fold it in half and it's ready to use.

```
TO TAG
REPEAT 2 [FORWARD 50 RIGHT 90
   FORWARD 125 RIGHT 90]
REPEAT 2 [FORWARD 50 LEFT 90
   FORWARD 125 LEFT 90]
END
```

Now you can use a shape to decorate your tag. Just choose a shape, pick the turtle's pen up and move it to where you want it on the tag and type STAMP.

You can add text to your tag simply by typing GWRITE or pressing <Open-Apple>-<W>. Move the cursor to the spot where you want to type. Use the arrow keys rather than the space bar to move across the tag outline and the shape that you have stamped. You can read a font from the Logo PLUS language disk (using the READFONT command) or even create your own font to use on the tags! Here are some sample tags that use a variety of shapes and fonts.



Happy holidays, Logo-style!

Dorothy Fitch
Director of Product Development
Terrapin Logo, 400 Riverside Street
Portland, Maine 04103

Cooperative Creations

by Jandy Bird

It seems that most recent educational reports, research, and pleas for reform call for an increased emphasis on cooperative learning and on problem solving. Logo users might be tempted to respond, "So what else is new?" The establishment of a "Logo environment" has both cooperative work and problem solving at its heart. This is easily seen in the work of Dan and Molly Watt, for instance, where problem solving strategies are developed and studied, and where children are often working in pairs on a project.

At grades K-4, my experience with cooperative learning using LogoWriter has been an energizing one. What follows is an account of three different experiences at elementary grade levels, employing cooperative learning to solve problems using LogoWriter.

First grade stories

To familiarize first graders with the graphics and word processing capabilities of LogoWriter, we first had the children do some simple typing and we taught them the turtle move steps, Turtle-Move keys (Apple-9), arrow keys, and ESC. Then came SETC and SETSH and stamp. After each student had some experience with these commands, we divided the class into groups of three. On a large monitor, we displayed the shapes page. (If you have no large monitor, each group could have a print of the shapes page).

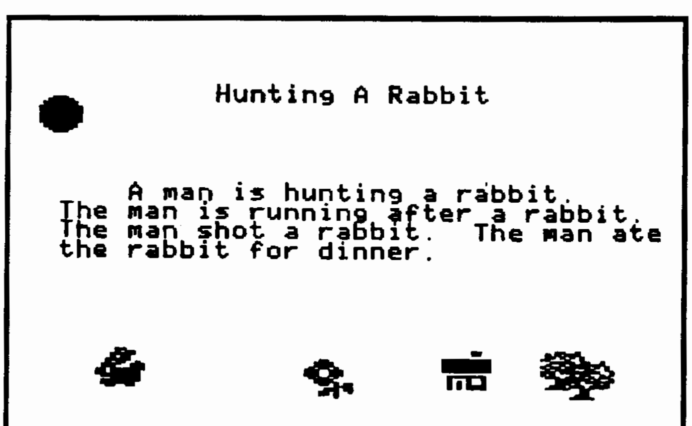
In the groups (not at the computer), each person in the group was told to choose one of the shapes from the shapes page. When each group member had a shape in mind, the group task was to think up a story using the three shapes. Then, with paper and pencil, each group was to draw a rough sketch of the shapes and the story ideas. We did this in a lab session with several volunteer parents as well as the classroom teacher and me, so there were plenty of hands to help each group with the writing and spelling of words. (If this is done with a single teacher, it might be wise to engineer this stage so the groups are not all planning their story at once.)

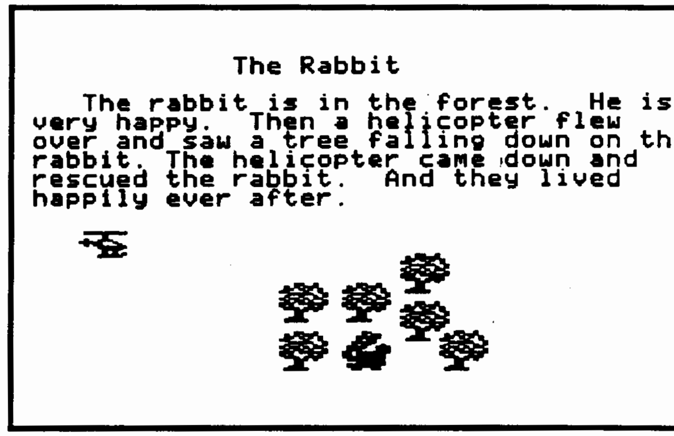
At the end of this first step in the process, each group had a paper with a plan for their story on it. In some groups, the plan was completely written out by the students themselves. In others, it was a sketch with only a few words. In cases where teachers grouped the students according to similar ability, some groups obviously had more elaborate plans than others. In other cases, where students were placed in groups randomly or were grouped with mixed ability, there were varied types of plans. The introduction, selection of shapes, and story plan took generally one lab session of about an hour.

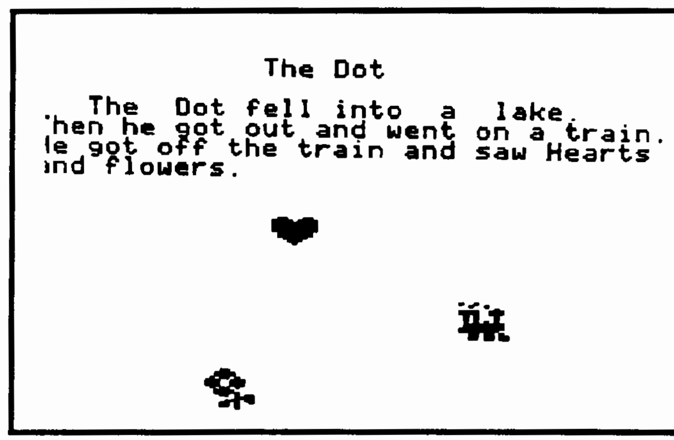
The next step was for each group to go to the computer and to enter their story. All three children in a group worked at one computer. This emphasized the cooperative nature of the project. Students in each group were urged to share the typing and to help each other with finding letters, using the shift key for capitalization, and proofreading. After the story was typed, the students set the turtle shape, moved the turtle around, and stamped the shapes on the page according to their plan. More advanced students had time to add details such as color or more shapes to the drawings. Using PRINTSCREEN, each story was printed three times (a copy for each student in the group), and some classes bound their stories into a class booklet. Some students also colored their stories by hand.

The final step in this process (important and fun as well) was sharing with the whole class. The use of a large monitor made this step particularly exciting for the children. (If you don't have a large monitor but have access to a VCR with a large screen TV you can use that instead.) The response to each group's effort was applause! Each story also offered practice in reading and a chance to discuss different graphic effects of LogoWriter which had been utilized.

Below are three examples of first graders' efforts. The "Hunting a Rabbit" group is obviously of a practical bent. "The Dot" was created by a group that wanted to use one of the empty shapes as well as three other shapes. The group that wrote "The Rabbit" combined a rescue story with careful placing of forest trees.







The students were very proud of their group's stories and were attentive to each others' work as well. The use of LogoWriter's word processing allowed for an attractive final product and for correcting mistakes more simply than recopying pages by hand. The excitement generated by this kind of approach was evident through the whole process as students worked cooperatively together, and teachers were pleased at the experiences with language and planning that this project provided.

Jan J. Bird, Ed.D
Conover Road School
80 Conover Road
Colts Neck, New Jersey 07722
201-946-8590
CIS: 73517,3270

Logo LinX

TURTLE OUT OF BOUNDS: Terrapins Off-Screen
by Judi Harris

Are there turtle fan(atic)s on your holiday shopping list? Do you find yourself receiving terrapin-related gifts from friends and family who know of your passion for Logo? You are not alone. It seems that a fondness for "pixelated" Logo turtles has engendered an attraction to the reptiles themselves for more than one Logo enthusiast. In this month's column, I'd like to share some non-electronic turtle resources.

Turtle Gifts

The Center for Marine Conservation (formerly the Center for Environmental Education) has worked to conserve marine habitats, prevent marine pollution, manage fisheries, and protect endangered marine species since 1972. It also publishes a delightful mail-order gift catalog that includes many sea turtle items, such as jewelry, toys, clothing, sculpture, and housewares. A catalog may be requested by writing to:

Whale Gifts 167-2 Elm Street P.O. Box 810 Old Saybrook, CT 06475

Credit card orders may be placed at any time by calling 1-800- 227-1929 (in Connecticut, 1-388-4436).

More information about the conservation efforts of this organization may be requested by writing to:

Center for Marine Conservation
1725 DeSales Street, NW Washington, D.C. 20036

Turtle Books

An electronic search of R.R Bowker's "Books in Print/ Reviews Plus" CD ROM (copyright 1988, Online Computer Systems, Inc.) revealed the following list of children's books about turtles or tortoises.

Ancona, G. (1987). *Turtle watch*. New York: Macmillan. ISBN #0-02-700910-6 (46 pp., photographs)

Asch, F. (1980). *Turtle tale*. New York, NY: Dial Books for Young Readers. ISBN #0-8037-8785-5 (32 pp., illustrated)

Beverly, D. E. (1986). *The tale of the wise little sea turtle*. New York, NY: Vantage Press, Incorporated. ISBN #0-533-05506-7 (27 pp.)

Blotnick, E. (none). *Blue turtle moon queen*. Berkeley, CA: California Street. ISBN #0-915090-20-1 (120 pp., illustrated)

Cosgrove, S. (1978). *Tee-Tee*. Los Angeles, CA: Price/Stern/ Sloan Publishers. ISBN #0-8431-1400-2 (28 pp., illustrated)

Craig, J. (1982). *Turtles*. Mahwah, NJ: Troll Associates. ISBN #0-89375-665-2 (32 pp., illustrated)

Cromie, W. J. (1970). *Steven & the green turtle*. New York, NY: Harper &Row Junior Books Group. ISBN #0-06-021374-4 (64 pp., illustrated)

Dodd, L. (1985). *The smallest turtle*. New York, NY: State Mutual & Book Periodical Service, Limited. ISBN #0-907349-35-8

Duff, M. (1981). *Dancing turtle*. New York, NY: Macmillan Publishing Company, Incorporated. ISBN #0-02-733010-9 (32 pp., illustrated)

Eager, E. (1986). *Magic by the lake*. San Diego, CA: Harcourt Brace Jovanovich, Incorporated. ISBN #0-15-250443-5 (156 pp., illustrated)

Edler, T. J. (1977). *Maurice the snake & Gaston the near-sighted turtle:* Tim Edler's tales from the Atchafalaya. Baton Rouge, LA: Little Cajun Books. ISBN #0-931108-00-4 (36 pp., illustrated)

Fine, E. H. (1988). *The turtle & tortoise*. Mankato, MN: Crestwood House, Incorporated. ISBN #0-89686-392-1 (48 pp., illustrated)

Gault, C. (1974). *Norman plays second base*. New York, NY: Scholastic, Incorporated. ISBN #0-590-03197-X

Gault, F. (1975). *Norman joins the football team*. New York, NY: Scholastic, Incorporated. ISBN #0-590-09931-0

Goldsmith, H. (1980). *Toto the timid turtle*. New York, NY: Human Sciences Press, Incorporated. ISBN #0-87705-525-4 (32 pp., illustrated)

Hadithi, M. (1988). Tricky tortoise. New York: Little, Brown. ISBN #0-316-33724-2 (30 pp., illustrated)

Holling, H. C. (1951). *Minn of the Mississippi*. Boston, MA: Houghton Mifflin Company. ISBN #0-395-17578-X

Jeffery, G. (1988). *Thomas the tortoise*. New York: Crown Books. ISBN #0-517-57043-2 (illustrated)

Johnson, S. A. (1976). *Penelope the tortoise*. Minneapolis, MN: Carolrhoda Books, Incorporated. ISBN #0-87614-072-X (24 pp., illustrated)

Lloyd, D. (1986). *Waldo the tortoise*. New York, NY: Scholastic, Incorporated. ISBN #0-590-33654-1 (32 pp., illustrated)

Oke, J. (1986). *The impatient turtle*. Elkhart, IN: Bethel Publishing Company. ISBN #0-934998-24-8 (110 pp., illustrated)

Oda, H. (1986). *The turtle*. Milwaukee, WI: Raintree Publishers, Incorporated. ISBN #0-8172-2572-232 (32 pp., illustrated)

O'Donnell, E. L. (1989). *I can't get my turtle to move*. New York: Morrow. ISBN #0-688-073239 (illustrated)

Oneal, Z. (1979). *Turtle & snail.* New York, NY: Harper & Row Junior Books Group. ISBN #0-397-31829-4
Reed, L. *Smiley the proud turtle.* Shreveport, LA: Archival Services, Incorporated. ISBN #0-910653-14-3 (64 pp., illustrated)
Riedman, S. R. (1974). *Turtles: Extinction or survival?.* New York, NY: Harper & Row Junior Books Group. ISBN #0-200-00126-4 (160 pp., illustrated)
Ross, D. (1986). *Tiny turtle's thanksgiving.* New York: Morrow. ISBN #0-688-06440-X (illustrated)
Schlachter, R. (1986). *Winter fun.* Mahwah, NJ: Troll Associates. ISBN #0-8167-0585-2 (48 pp., illustrated)
Serventy, V. (1987). *Turtle & tortoise.* New York, NY: Scholastic, Incorporated. ISBN #0-590-40228-5 (24 pp., illustrated)
Dr. Seuss. (1958). *Yertle the turtle & other stories.* New York, NY: Random House, Incorporated. ISBN #0-394-80087-7
Sisters of the Community of Jesus Staff. (none). *Myrtle the turtle.* Orleans, MA: Paraclete Press. ISBN #0-941478-84-X (32 pp., illustrated)
Smith, W. (1977). *Timothy turtle learns about love.* Marina del Rey, CA: DeVorss & Company. ISBN #0-87516-244-4
Stevens, G. (1985). *The smallest turtle.* New York: Goldstar. ISBN #0-918831-07-5 (illustrated)
Stevenson, J. (1979). *Fast friends.* New York, NY: Greenwillow Books. ISBN #0-688-80197-8 (64 pp., illustrated)
Troughton, J. (1986). *How rabbit stole the fire: A North American Indian folk tale.* New York: Harper. ISBN #0-87226-040-2 (illustrated)
Troughton, J. (1986). *Tortoise's dream: an African folk tale.* New York: Harper. ISBN #0-87226-039-9 (illustrated)
Valli, V. (1986). *Aristotle & I: Memoirs of a Greek tortoise.* New York, NY: Vantage Press, Incorporated. ISBN #0-533-06785-5 (88 pp.)
Waters, J. F. (1972). *Green turtle mysteries.* New York, NY: Harper & Row Junior Books Group. ISBN #0-690-35994-2
Waters, J. F. (none). *The hatchlings: A book about baby turtles.* New York, NY: Walker & Company. ISBN #0-8027-6362-6
Weiss, E. (1983). *The angry book starring temper tantrum turtle.* New York, NY: Random House, Incorporated. ISBN #0-394-85541-8 (32 pp., illustrated)
Williams, B. (1974). *Albert's toothache.* New York, NY: Dutton, E. P. ISBN #0-525-45037-8 (32 pp., illustrated)
Williams, B. (1977). *El dolor de muelas de Alberto.* New York, NY: Dutton, E. P. ISBN #0-525-29168-7
Ziefert, H. (1987). *Where's the turtle?* New York: Harper. ISBN #0-694-00183-X (illustrated)

Turtle Poems

Regional and city libraries often have literature indexes that can be consulted to find turtle folktales and poems. The following list of poems found in many popular poetry anthologies was collected in this way, and can be consulted to provide terrapin-related study for older students.

H. Asquith, "The Tortoise."
E. Dickinson, "Too Much Proof Affronts Belief."
J. T. Fields, "Song of the Turtle and Flamingo."
R. Frost, "The Egg and the Machine."
A. A. Knipe, "A Discovery."
D. H. Lawrence, "Tortoise Shout."
V. Lindsay, "The Little Turtle."
W. de la Mare, "Old Shellover."
H. Wing, "Turtle Town."
E. Wylie, "The Tortoise in Eternity."

My favorite turtle poem was written by Ogden Nash. In an amusing way, it expresses the wonder that I feel about the instructional fertility of such a seemingly simple notion as the turtle.

The turtle lives 'twixt plated decks
Which probably conceal its sex.
I think it clever of the turtle
In such a fix to be so fertile
(Ogden Nash, "The Turtle")

It seems that powerful ideas are expressed in many forms.

Judi Harris
621F Madison Avenue
Charlottesville, VA 22903
CIS: 75116,1207
BitNet: jbh7c@Virginia

MathWorlds

CLIME Report: A problem, a plea, and some activities
by Ihor Charischak

Some Ominous Clouds on the Technology Horizon

In my last Clime article (*Logo Exchange*, May, 1989) I mentioned that I was about to leave for the National Council of Teachers of Mathematics (NCTM) Annual conference in Orlando, Florida where CLIME (The Council for Logo in Mathematics Education) would be officially recognized as an affiliated group of NCTM. Well, that is now official and I am happy to announce that a Logo group is now recognized as an official member of the mathematics community. But not all is well in Mathland. At the conference I discovered that of the 633 sessions (over a period of three days and attended by about 5000 math teachers), there were only 5 Logo sessions. This total is down from 17 in 1988 and 25 in 1987. Even worse (as George Bright, our NCTM representative informed me) was that there were only 71 sessions that had a technology theme! This was particularly surprising in light of the fact that NCTM was celebrating their newly established Standards for School Mathematics which is intended to serve as a model for mathematics education into the 21st century. (The full name is Curriculum and Evaluation Standards for School Mathematics and is available from the NCTM office in Reston, Virginia.) Since this was so hard for me to believe, I looked for and found some conference organizers to speak to who might be able to shed some light on why this was so. But they were just as surprised at my statistics as I was and they assured me that there was no conspiracy on the part of NCTM to torpedo computers in mathematics education. They suggested that I contact the chairperson or persons of future conferences and share my concerns. Now it was already too late to do anything about the 1990 conference in St. Lake City – that program is pretty well set. But there is still hope that something can be done about the 1991 conference in New Orleans. I have written a letter to the program committee sharing my concerns. (Write to me at the address at the end of this column if you would like a copy of my letter.)

The Plea

There are things you can do as well to make a difference. You can speak at regional, local, or national math teachers meeting. There are plenty of new as well as veteran teachers that would love to hear more about how Logo can help in teaching mathematics. If you wish a list of future NCTM meetings and a speaker recommendation form (you can recommend yourself), contact NCTM (1906 Association Drive, Reston, VA 22091) and they will be happy to send the information. (The deadline for the New Orleans meeting is November 15.)

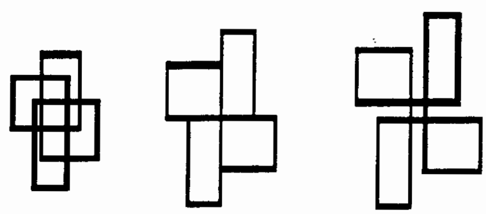
Some Activities

Last spring, CLIME declared that it would become a clearinghouse for interesting math microworlds. Our spring issue of the CLIME News discussed microworlds and included several examples that I think definitely captured the spirit of the NCTM Standards. (See examples that follow.)

CLIME has made these microworlds available on disk (Microworlds Disk #1). If you wish to become a member of CLIME ($10 North America, $15 elsewhere) and/or get the CLIME Microworld's Disk #1 write to the address given at the end of this column.

Spiros
by Bob Jensen

S is a command that takes three numbers as inputs. It then draws a right spirolateral by consecutively walking off each distance inputted followed by a 90 degree right turn. This pattern is continued until the path closes. The resulting figure is then classified as either a WUMPUS, WHIMSY, or GLOOP.



Wumpus? Whimsy? Gloop?

Can you predict what figure will result based on the numbers inputted? (The procedures for this activity are on the CLIME Microworlds Disk #1)

Exploring IT: A Microworld for Discovering Division
by Richard Binswanger

This microworld is described in more detail in Richard's article in the *Arithmetic Teacher* (Dec., 1988).

The IT procedures are:

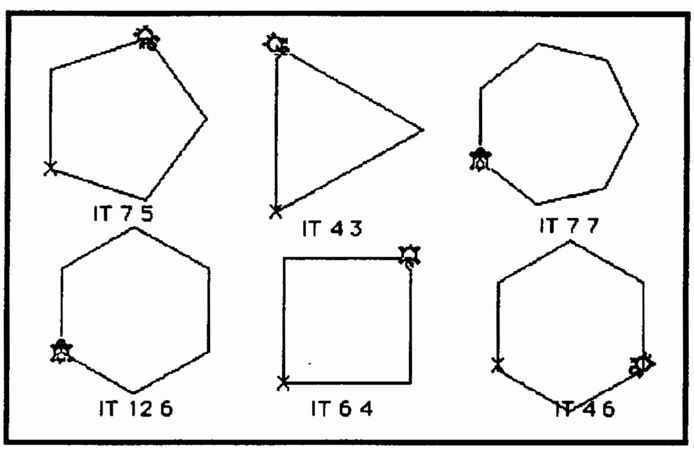
```
to it :m :n
x
repeat :m [forward 300/:n wait 4
   right 360/:n]
end
```

```
to x
right 40
forward 5
back 10
forward 5
left 80
forward 5

back 10
forward 5
right 40
end
```

The input M is the total number of segments the turtle draws, while N describes the number of sides in the regular shape that the turtle draws. For example

```
IT 15 5
```

draws a pentagon where the turtle retraces the pentagon three times.



Some questions to explore:

- Predict what the turtle will do when you give some inputs to IT. For example, what will IT 9 4 draw?
- What figure does the turtle draw?
- Where on the figure will the turtle wind up?
- How many complete revolutions did the turtle make around the shape?
- What values of :M and :N should be used to have the turtle make six revolutions around a hexagon?
- What values of :M and :N will cause the turtle to finish where it starts? For that matter, list all values of :M that will cause the turtle to land on a particular vertex for a given :N.

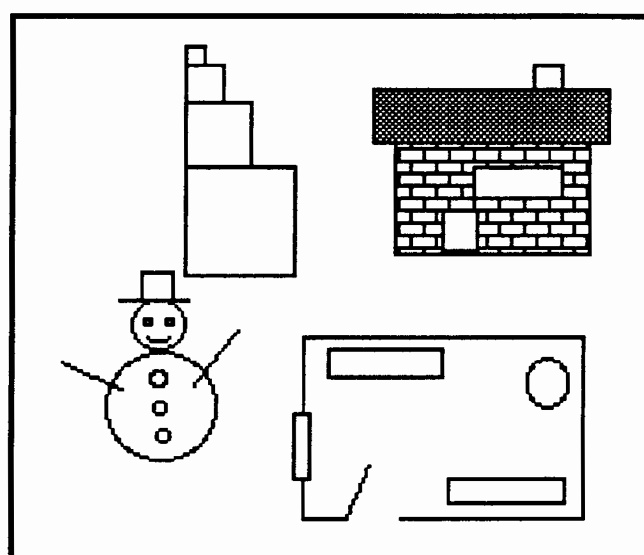
Explore repeating IT. For example, REPEAT 5 [IT 8 5] places an X at every vertex of the pentagon, whereas REPEAT 6 [IT 8 6] only places an X at vertices 0, 2, and 4, although it draws each one twice. When using REPEAT :M [IT :M :N] with arbitrary values, which vertices will have Xs and which won't? Furthermore, in what order will these Xs get drawn?

Geo Shapes
by Ihor Charischak

This is a simple microworld that offers students an opportunity to use shapes in creating designs on the computer screen and on paper.

I ask the students to draw some pictures using a set of procedures (below) which I demonstrate to them. What follows depends on their requests. Some of the questions that turned into problem solving activities were:

- How can I add color to my picture?
- I need some 6 sided shapes. What are they called?
- How do you draw a star?
- Can you make a circle and a square that are the same size? (What the student meant to ask was: How do I inscribe a square inside a circle? But I like the original question and I have asked it on occasion). Δ



Math Worlds--continued

Useful procedures:

SQUARE *length*	draws a square with sides equal to *length* turtle steps
TRIANGLE *length*	draws an equilateral triangle with sides equal to *length*
RECTANGLE *length width*	draws a rectangle with dimension *length and width*
CIRCLE *circumference*	draws a circle with a circumference of *circumference*
MOVE	allows user to move turtle with arrow keys

Ihor Charischak
President
CLIME
10 Bogert Avenue
White Plains, NY 10606

Logo and Company

Sight and Sound: The Next Decade
Part 1 - The Sounds of Tomorrow
by Glen L. Bull and Gina L. Bull

Logo was developed long before personal computers were available. Some of the early development occurred before 1970, but personal computers did not become widely used in the schools until the 1980's. A visitor to the Logo development laboratories in the early 1970's was, in a sense, making a trip to the future. In our last column before the next decade it seems appropriate to revisit classroom computing of the future.

In his introduction to "The Twilight Zone," Rod Serling described a journey involving the dimensions of "sight and sound." Tomorrow's computers will have greatly enhanced capabilities in those two dimensions. Speech and enhanced graphics will be a standard feature of classroom computers of the next decade. In this month's column we will explore the auditory dimension. Next month we will turn our attention to visual media.

Charts describing educational computing sometimes show a pyramid in which the least powerful computers are found in the lower grades and more powerful computers are found in high schools and universities.

university - NeXT & Sun Workstations
high school - IBM PS/2 or Macintosh Computers
middle school - Apple II GS (Graphics & Sound) Computers
elementary school - Apple II+ and IIe Computers (64 to 128 Kilobytes)

Despite the charts, the most powerful computers are most needed in the lower grades. If you consider that in kindergarten and first grade children are just beginning to read, availability of speech and realistic graphics is even more important than it is in later grades. Recently Helen Edens of the Virginia Department of Education and Bruce Gansneder, a researcher at the University of Virginia, completed an extensive survey of Virginia's schools and obtained some interesting results. The data are still being analyzed, but preliminary results suggest that elementary and middle school teachers as a group are likely to use computers more extensively and more routinely than high school teachers in every area but word processing. Powerful computers with speech and sound are needed in the lower grades, and teachers in those grades are likely to have the expertise to make use of these capabilities.

Paradoxically, more powerful microcomputers are also easier to use. Increased speed and memory makes them more responsive, and makes it possible to install graphic user interfaces and icons that facilitate ease of use. On the Macintosh, for example, a subdirectory is opened by clicking on a picture of a file folder. This metaphor is so clear that even a six-year-old can master it. On the NeXT computer this imagery is extended by opening the file folder icon as a document is dropped into it. The OS/2 operating system on IBM and IBM-compatible computers will also have a graphic user interface. In contrast, anyone who has ever attempted to explain pathnames in MS-DOS or Pro-DOS knows the concept is far from intuitive.

Computers can be given the gift of speech in two ways - through *speech synthesis* or by means of *digitized speech*. There are advantages and disadvantages to both approaches. However, each approach is more easily achieved on more powerful computers. A *speech synthesizer* converts printed text on the screen of the computer. The advantage of this method is that anything that can be typed on the keyboard of the computer can be spoken. This is a significant advantage because it allows children to type their own stories and hear them read aloud. The disadvantage of speech synthesis is that it has a robotic quality. This sometimes distresses adults, but children do not mind since they expect a computer to sound like a robot.

A *sound digitizer* turns the computer into a *digital tape recorder*. Phrases spoken into a microphone connected to the computer are stored in the memory of the computer. The advantage of this method is that the speech has a very natural quality, just as though it were recorded on a tape recorder. Also, it is possible to record other sounds, such as the buzzing of a bee. The disadvantage of digitized speech is that it consumes large amounts of storage in the computer, so it is not practical to store large numbers of phrases. Also, the sound can only be played back if it has been previously recorded, so this method is less flexible.

Method	Advantage	Disadvantage
Speech Synthesis	Flexibility	Robotic Quality
Digitized Speech	Natural Quality	Limited Number of Phrases

Speech Synthesis and Logo

It is perfectly possible to create a talking version of Logo on an Apple II or IBM computer through use of a speech synthesizer. A serial printer such as an Imagewriter is unplugged and replaced with an external speech synthesizer. Thereafter text sent out the printer port to the serial cable is spoken by the speech synthesizer rather than printed on a sheet of paper. The computer is not aware that the printer at the other end of the cable has been replaced by a speech synthesizer. When the speech synthesizer receives the text transmitted by the computer, it is programmed to say each word. The Votrax and Echo GP (general purpose) external speech synthesizers work well for this purpose.

Once the speech synthesizer has been attached to the computer, a short procedure Say is added to Logo. Say sends the text out the printer port, where it is spoken by the speech synthesizer. For example, if the speech synthesizer is attached to a serial card in Slot 1 of an Apple II computer, the Say procedure would be written in the following way in Apple Logo II. (The DRIBBLE command sends text printed on the screen to the specified file or device.)

```
TO Say :Words
DRIBBLE 1
PRINT  :Words
NODRIBBLE
END
```

Many Logo-like programs such as *Talking Textwriter* (a talking word processor for Apple and IBM computers sold by Scholastic) also use speech synthesizers. Most of these programs use the Echo II speech synthesizer card sold by Street Electronics which plugs into an internal slot in the Apple or IBM computer. Unfortunately the Echo internal speech synthesizer is not compatible with Logo; evidently some of the memory used by this internal card overlaps with memory needed by Logo. A student hired by Street Electronics worked on this problem one summer, but did not identify a solution before returning to school. We have never been able to identify a way to make any version of Logo on the Apple operate the internal Echo speech synthesizer card, even though it is the most popular and most widely sold speech synthesizer card for that computer. Therefore a teacher who wants to use both talking Logo and Logo-like programs such as *Talking Textwriter* is faced with the uneconomical prospect of purchasing both an internal and an external speech sythesizer. (If a programming genius has discovered a way to make Logo work with the internal Echo card, we would very much like to hear about it.)

Digitized Speech and Logo

Speech digitizers are also available for the Apple II computer. However, the Apple II 6502 microcprocessor is comparatively slow in comparison with more modern microprocessors, and the capacity of a single-sided floppy disk is limited in comparison to a hard disk with megabytes of storage. We know of no one who has added digitized speech to Apple versions of Logo.

David Cartmell of the IBM Corporation has demonstrated digitized speech for IBM Logo at a number of Logo conferences. Dave developed a digitized speech editor named Phred (Phrase Editor) which can be used to record and edit digitized speech phrases. These digitized phrases can then be replayed with Logo. Unfortunately IBM never made this extension to Logo available as a commercial product, and, in fact, has never updated IBM Logo past Version 1.1. Use of digitized speech with IBM Logo also requires an IBM speech card.

Limitations of Current Classroom Computers

It is clear that the addition of speech to classroom computers is invaluable, especially in the lower grades. The reason use of speech is not more widespread is also clear – today's classroom computers are not sufficiently powerful. Even so, use of speech with current computers in the classroom is feasible, but cost and ease of use are limiting factors.

<u>Economics</u>

Inertia always leads to a tendency to use what is at hand. BASIC is more widely used than Logo because it comes with the computer (Applesoft BASIC with Apple computers and Microsoft BASIC with IBM computers). Speech on current classroom computers requires the addition of a speech synthesizer or digitizer. Not only does this require additional initiative on the part of the teacher, it is an added expense which must be justified within the school system. In the decade to come, more powerful classroom computers will have built-in speech synthesis and digitization capabilities.

<u>"Standard" Logo?</u>

Teachers who use Logo have more initiative than the average teacher. If you are using Logo with your class, you have found the funds to purchase Logo for your class rather than accepting the standard programs that came with the computer. Then you took the extra step of learning to use Logo, often on your own.

However, there's another factor that limits the widespread use of speech with Logo, beyond the extra initiative required to acquire an additional piece of hardware. The problem is that there's no standard way to access the printer port. The method is different in every dialect of Logo, and some early versions of Logo did not provide any means of all.

- Terrapin Logo provides one type of printer command (**OUTDEV**).
- Apple Logo does not provide any direct means of accessing the printer.
- Apple Logo II provides a different command than Terrapin (**DRIBBLE**)
- IBM Logo uses the **DRIBBLE** command in a slightly different way than Apple Logo II (**DRIBBLE 1** versus **DRIBBLE "COM1**)
- Although it is possible to access the printer port directly in LogoWriter, the means for doing this is not documented in the user's manual.

This list could be extended for other versions of Logo. The fact that the same command has a different form in every version of Logo inhibits use of talking versions of Logo.

The differences mean that knowledge gained by one teacher can not be shared with teachers using other dialects of Logo. In the story of creation described in Genesis it said that originally "the whole earth was of one language, and of one speech" before the Lord said "confound their language, that they may not understand one another's speech." The stated purpose for disrupting the shared language was to prevent men from working cooperatively together on the Tower of Babel. In similar fashion, the language of Logo has been confounded so that teachers may not understand one another's programs, inhibiting sharing and cooperative efforts. (This is why you may sometimes hear teachers today speak of the "confounded Logo language.")

Commands which are different in every version of Logo combined with potential problems with interface cables and switch settings on the serial port and the synthesizer itself all conspire to make the teacher who uses a talking version of Logo more the exception than the rule.

Future Computing: Speech Synthesis

As computers become more powerful, the capability for speech synthesis will be inherent in the computer. As a case in point, every Macintosh has a built-in capacity for speech synthesis without any additional hardware. Since the capacity is built into the computer, there are no cables or switch settings to configure.

A standard software package, MacinTalk, can be used to access the built-in speech synthesis capabilities of the Macintosh. MacinTalk is widely available through user groups and computer bulletin boards. MacinTalk is also available through the Apple Programmers and Developers Association (APDA). Membership in APDA is $25, while the cost of MacinTalk (Version 1.31) to APDA members is only $10.

MacinTalk provides the basic speech synthesis code, and is placed in the Macintosh system folder. It can be used as the heart of a number of different user interfaces. A number of programs such as HyperMacintalk by Dennis C. DeMars are available for free, non-commercial distribution and use. There are two differences between today's classroom computers and computers of the next decade.

- On more powerful computers of the future, speech synthesis capabilities will be built into the computer. With additional processing power, functions that were formerly accomplished through hardware will in some instances be accomplished through software.

- Provision of an inherent capacity that comes with every computer will make a standard interface possible, removing dialectical differences that formerly separated them and enabling users to share work.

Dennis DeMars can share HyperMacintalk secure in the knowledge that every person who has a Macintosh will be able to use his program. (Hypercard is required to use HyperMacintalk, but a copy of Hypercard is provided with every Macintosh.) Further, a button for installation of a **SAY** command in user's own Hypercard programs is provided with HyperMacintalk. This permits users to use Dennis DeMars programs directly, or use his **SAY** command as the basis for their own programs.

Recently we wanted to create a program which would allow first-graders to experiment with word families. A word family is group of words which have different initial or final sounds, but common middle sounds:

C	AN	C	AT
B	AN	B	AT
M	AN	M	AT

We created a series of electronic notecards in Hypercard. Each electronic card contained a different word family. When the user presses a key on the keyboard, the letter at the beginning of the word changes, and the computer says the new word out loud. It only took a few seconds to develop this program; the Hypercard script for this program is only six lines long:

Logo & Company--continued

```
On Idle
  Put inkey() into it
  If chartonum of it is not 0 then
    Put it into card field Letter
    Say it & card field
    WordFamily & "."
  End if
  Pass idle
End Idle
```

We could have as easily developed this program in Logo. The procedures would have been just as easy to write. However, to make the program available, it would have been necessary to develop different sets of instructions for:

- different versions of Logo
  (Terrapin, Apple Logo II, LCSI Logo II, Logowriter, etc.)

- switch settings on different types of speech synthesizers
  (Votrax, Echo, etc.)

- switch settings on different serial interface cards
  (Super Serial card, etc)

In the end, fewer than ten percent of teachers with Apple II computers would have had the required hardware necessary to run the program in any event. In contrast, the same program developed in Hypercard will run on any Macintosh with MacinTalk installed in the system folder.

This example is not described to persuade you to exchange your Apple II or MS-DOS computer for a Macintosh. Rather it is to provide a glimpse of the future that may be available when speech becomes a standard, easily accessed feature on all classroom computers.

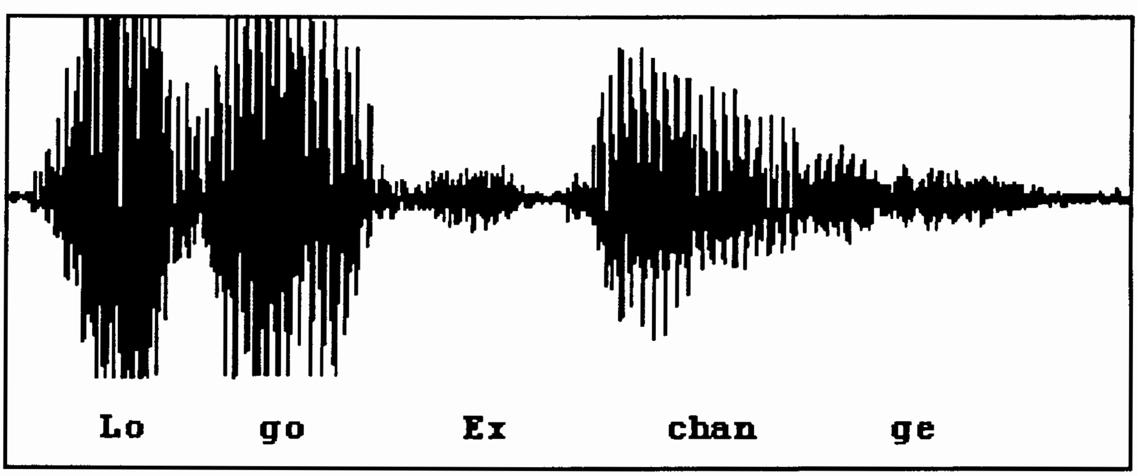
Future Computing: Digitized Speech

The capacity for digitized speech will also be built into future computers or available at very little cost. For example, a popular sound digitizer for the Macintosh, MacRecorder, is available from mail order firms such as MacConnections for about $175. MacRecorder is the successor to programs such as Dave Cartmell's PhraseEditor. With the increased power of faster processors, it is possible to acquire, edit, and store digitized speech with greater fidelity and ease of use. MacRecorder also makes it possible to place a recorded phrase in any Hypercard program. MacRecorder is required to digitized the phrase, but once it is captured it the phrase can be replayed on any Macintosh computer.

The visual display of the phrase "Logo Exchange" is shown below.

Other computers such as the Steve Job's NeXT computer have sound digitization capabilities built in. This makes it possible to send an electronic mail message to another user with a NeXT computer, and attach a spoken recording to a written document. When the user receives the message, the recording can be played back as the written document is read.

Digitized speech requires a considerable amount of storage space. The phrase "Logo Exchange" displayed below was 1.14 seconds long, and consumed 22K (kilobytes) of disk space. However, considerable is a relative term. The standard floppy disk of an Apple II computer holds about 140 kilobytes. The optical erasable disk that is standard on every NeXT computer stores a quarter of a gigabyte. The educational price of an Apple IIGS computer is currently about $1500 to $2000, depending on configuration. The educational price of a NeXT computer is currently about $6000. Although the NeXT computer costs four times as much, its disk holds more than 2,000 times as much as the Apple II computer. This means that



if phrase such as "Logo Exchange" requires 25,000 bytes of storage on average, it would be possible to store 4 phrases on an Apple II disk and 10,000 phrases on the NeXT disk.

Computer	Price	Bytes Storage	Phrases
Apple II	$1500	100,000	4
NeXT	$6000	250,000,000	10,000

In short, digitized speech is a practical tool on the NeXT computer. When these capabilities become standard in the classroom, it will be possible to use digitized speech and sounds to accompany text.

Summary

As we enter the next decade it is evident that Rod Serling's "dimension of sound" will enter the classroom. In next month's column at the beginning of the new year and new decade we will consider the dimension of sight in "Part II: Visions of the 90's."

Glen and Gina Bull
Curry School of Education
Ruffner Hall
University of Virginia
Charlottesville, VA 22903

BITNET addresses:
Glen: LB2B@ VIRGINIA. Gina: RLBOP@VIRGINIA.

A New Primitive for LogoWriter

by Charles E. Crume, B.S

The Logo computer language differs from many other computer languages in that it provides the capability to work with words and lists. (A word is defined to as one or more characters with no intervening spaces and a list is defined as one or more words with intervening spaces.) Logo words and lists are dynamic entities — the number of elements they contain can change throughout the execution of a program.

To facilitate working with words and lists, Logo provides primitives such as FIRST, BUTFIRST, LAST, BUTLAST, and ITEM. As we will see, the primitives BUTFIRST and BUTLAST perform the opposite function of FIRST and LAST RESPECTIVELY. Most versions of Logo, however – including *LogoWriter* – do not provide a primitive to perform the opposite of ITEM (a function I often need). Therefore, I decided to write such a primitive. The procedure is called BUTITEM. The code for this primitive and an example program using it are shown at the end of this article. Before presenting BUTITEM however, a review of the primitives FIRST, BUTFIRST, LAST, BUTLAST, and ITEM might be helpful.

The primitive FIRST reports the first element of its input. For example, the command:

```
PRINT FIRST [THE WEATHER IS FINE
   TODAY]
```

prints the word

```
THE
```

In instances where everything except the first element of the input is needed, the primitive BUTFIRST is used. For example, the command:

```
PRINT BUTFIRST [THE WEATHER IS FINE
   TODAY]
```

prints the list

```
[WEATHER IS FINE TODAY]
```

Sometimes however, the last element instead of the first element is needed. In such a case the primitive LAST is be used. For example, the command:

```
PRINT LAST [THE WEATHER IS FINE
   TODAY]
```

A New Primitive for LogoWriter--continiued

prints the word

```
TODAY
```

Just as the opposite of FIRST is BUTFIRST, the opposite of LAST is BUTLAST. To report everything except the last element of the input the primitive BUTLAST is used. For example, the command:

PRINT BUTLAST [THE WEATHER IS FINE TODAY]

prints the list

```
[THE WEATHER IS FINE]
```

In addition to these four primitives, Logo also provides the primitive ITEM (which reports any specific element contained in a word or list). For example, the command:

```
PRINT ITEM 3 [THE WEATHER IS FINE
   TODAY]
```

prints the word

```
IS
```

and the command:

```
PRINT ITEM 4 [THE WEATHER IS FINE
   TODAY]
```

prints the word

```
FINE
```

As mentioned earlier in this article, most versions of Logo do not provide a primitive that performs the opposite of ITEM. Thus, a procedure that reports everything except the specified item was developed. The procedure is called BUTITEM and the code is shown below:

```
TO BUTITEM :MYITEM :MYLIST
IFELSE :MYITEM = 1 [OUTPUT BUTFIRST
   :MYLIST] [OUTPUT SENTENCE FIRST
   :MYLIST BUTITEM :MYITEM - 1 BUT-
   FIRST :MYLIST]
END
```

The procedure BUTITEM takes two inputs. The first input is a positive integer specifying which element is to be removed. The second input is the word or list to be processed. For example, the command:

```
PRINT BUTITEM 3 [THE WEATHER IS FINE TODAY]
```

prints the list

```
[THE WEATHER FINE TODAY]
```

The following Logo program demonstrates how one could make use of the BUTITEM procedure:

```
TO DEMO
MAKE "PEOPLE [MOM DAD BROTHER SISTER
   AUNT UNCLE]
DISPLAY :PEOPLE
END

TO DISPLAY :X
IF EMPTY? :X [STOP]
MAKE "I (RANDOM COUNT :X) + 1
PRINT ITEM :I :X
DISPLAY BUTITEM :I :X
END
```

The above program will display the various people in a random order each time the program is executed.

Additional information on this and other Logo topics can be found in *Computer Science Logo Style, Volume 1: Intermediate Programming*, written by Brian Harvey, and published by the MIT Press.

Charles E. Crume, B.S.
Technical Consultant
University of Nevada System Computing Services
University of Nevada Reno

Centering On Logo

The 1989 International Computer Problem Solving Contest: Junior Logo Results

by
Donald T. Piele and Sharon Yoder

The International Computer Problem Solving Contest is an annual contest for precollege students conducted at local contest sites throughout the world. Held on the last Saturday in April, it challenges teams (from one to three members each) to solve five problems in two hours using any programming language. Last spring, approximately 300 teams participated in the Junior Logo Division (Grades 7-9). This month we present the results of the Junior Logo Division along with the problems and sample solutions. The Senior and Elementary Logo division results will be presented in coming months. The BASIC and Pascal contest results appear in *The Computing Teacher*.

From 1981 to 1988, the contest was sponsored by the University of Wisconsin-Parkside. Currently it is sponsored by ICPSC – a non-profit organization with financial support from USENIX, the technical association of UNIX users.

Introduction

For the third year in a row, teams from contest sites in Limerick, Dublin, or Cork in Ireland have won first place in at least one of the Logo Divisions of the International Computer Programming Contest. And for the second straight year, Anne Chazarreta, from the Cork Centre at Colaiste an Spioraid Naoimh, captured a first place in the International Computer Problem Solving Contest. She took top honors in the Junior Logo Division, followed by classmates from Cork ranking second, third and fifth.

Why do so many of the best teams in the Logo contest come from Ireland – a country in physical size less than 1% of the United States? When I asked this question of Michael Moynihan, the contest director at Cork, he offered this explanation:

> Five years ago, Dr. Fred Klotz initiated a series of courses involving Logo and mathematics for student teachers and in-service teachers enrolled at St. Patrick's College. He and Dr. Sean Close felt that Logo was an ideal vehicle for exploring mathematics. They also felt that mathematically able children could, at their own pace, explore mathematical concepts through Logo. As this would not be possible within the normal classroom situation, they decided to ask experienced teachers, who had access to a number of computers, to set up centres in their own areas. This is how our centre in Cork came about.
>
> As a result, Logo courses for mathematically able children have been held at twelve centres around Ireland. These courses are coordinated by Dr. Sean Close of St. Patrick's College of Education in Dublin. (Fred died tragically from injuries suffered in a cycling accident on April 1, 1988.) The children who attend these Logo courses are tested for a high proficiency in Mathematics and are aged from 9 to 12 years old. The courses are divided into two terms of ten weeks (two hours a week.) At the Cork Centre, instruction in Logo includes two dimensional graphics, words, and lists. This year, for the first time, we organized our own regional Logo Competition. The first Irish National Logo Contest was held in Dublin in 1988. Children from all of the Logo centres around Ireland come together in June to solve problems. The overall winner is presented with the Dr. Fred Klotz Memorial Trophy presented by the Computer Society of Ireland.
>
> We were absolutely thrilled that our teams achieved such distinction and that Anne Chazarreta was again our winner. We knew that our teams would be ranked, but, at this remove, we had no idea of the standard to be met. Consequently, we are delighted, but amazed, that we did so well.

Anne, aged 13, is now attending Secondary school. Mathematics is her favorite subject and she will be studying more next year along with English, Irish, History, Geography, Science, French, and Commerce and Home Economics. At this point she plans to study Medicine or become an veterinary surgeon.

Mr. Moynihan writes,

> Anne has a great love of animals and has strong opinions on animal rights. She feels strongly that live animals should not be used in experiments or be hunted for their fur, etc. In her spare time, she and her friends visit the animal home of the Cork Society for the Prevention of Cruelty to animals. Of course, like all teenagers, she enjoys listening to pop music.

Anne was one of eight teams to solve all five problems within the two hour time limit. Five of these teams came from the same contest site at Colaiste an Spinoraid Naoimh, Bishopstown, Cork, Ireland.

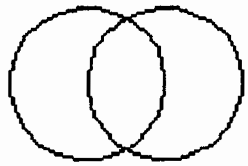
Junior Logo Division Problems,Sample Solutions and Comments

Grading the ICPSC problems is both fun and frustrating. It's wonderful to see what young people can do with their knowledge of Logo in a very short time. On the other hand ranking a large number of excellent solutions is extremely difficult. The difference between the rankings is often very small and it is especially difficult to leave out of the top rankings some excellent solutions.

This year as last, the final decisions were often based on style. Programming style includes both the formatting of the code, good use of modularity, and generally good programming logic. Sometimes the higher ranking went to a particularly sophisticated or creative solution, but usually it was just a matter of the overall style used.

If you are a teacher who plans to have your students participate in ICPSC next year, then you shouldn't expect that a "crash course" in programming style a few weeks before the contest wil produce winning entries. Under the time pressure of the contest, most students will revert to their "old ways" and forget issues of programming style. If they have been taught to use good programming style throughout the school year, then by the time the contest rolls around, it will be second nature and they can focus on the logic needed for solving the problems.

Below are the problems from the 1989 contest along with a possible solution. Look carefully at the programming style used. It will give you an idea of what is expected of students writing outstanding solutions to these problems.



1. TWO CIRCLES

Write a program that will produce this design. Note that each circle goes through the center of the other circle.

Note: Use $\pi = 3.14159$.

<u>Solution:</u>

```
TO TWO.CIRCLES
SET.UP
CIRCLE
MOVE.TO.SECOND
CIRCLE
END

TO SET.UP
CG
HT
END

TO CIRCLE
REPEAT 36 [FORWARD 5 RIGHT 10]
END

TO MOVE.TO.SECOND
RIGHT 90
PENUP
FORWARD RADIUS
PENDOWN
LEFT 90
END

TO RADIUS
OUTPUT ( 36 * 5 ) / ( 2 * 3.14159 )
END
```

Centering On Logo--continued

2. DOWN UP
Write a program that accepts a word and prints it in a pattern like this

```
LOGO
LOG
LO
L
L
LO
LOG
LOGO
```

Test your program with the words "LOGO and "SOLUTIONS

Solution:

```
TO DOWN.UP :WORD
PRINT :WORD
IF (COUNT :WORD) >1 [DOWN.UP BUTLAST
    :WORD]
PRINT :WORD
END
```

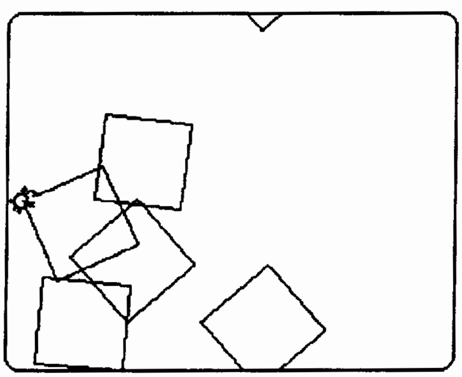
3. RANDOM SQUARES
Write a program that asks the user how many squares s/he would like to see on the screen. Your program should then place each square at a random spot on the screen. Thus, if you type

```
RANDOM.SQUARES
```

You see the question

```
HOW MANY SQUARES?
```

If you type the number 5, your output might look like this:



Run your program twice, using first 3 and then 5 as input.

Solution:

```
TO RANDOM.SQUARES
SET.UP
HOW.MANY
END

TO HOW.MANY
PRINT [HOW MANY SQUARES?]
REPEAT READWORD [PLACE.SQUARE]
END

TO SET.UP
CG
ST
CT
END

TO PLACE.SQUARE
PENUP
RIGHT RANDOM 360
FORWARD RANDOM 100
PENDOWN
SQUARE
END

TO SQUARE
REPEAT 4 [FORWARD 50 RIGHT 90]
END
```

4. COUNT LIST
Write a program that accepts a list of numbers and counts how many positive numbers and how many negative numbers are in the list. For example, if you type

```
COUNT.LIST [1 -5 -3 4 7 8 0 -2]
```

your program should print

```
THERE ARE 4 POSITIVE NUMBERS
THERE ARE 3 NEGATIVE NUMBERS
```

Test your program on the list given above.

Solution:

```
TO COUNT.LIST :THE.LIST
COUNT.THEM 0 0 :THE.LIST
END
```

```
TO COUNT.THEM :POS :NEG :LIST
IF EMPTYP :LIST
   [PRINT.OUT :POS :NEG STOP]
IF ( FIRST :LIST ) > 0
   [MAKE "POS :POS + 1]
IF (FIRST :LIST) < 0
   [MAKE "NEG :NEG + 1]
COUNT.THEM :POS :NEG
   ( BUTFIRST :LIST )
END

TO PRINT.OUT :NO.POS :NO.NEG
( PRINT [THERE ARE] :NO.POS
   [POSITIVE NUMBERS] )
( PRINT [THERE ARE] :NO.NEG [NEGATIVE
   NUMBERS] )
END
```

5. COIN FLIP

Write a program that simulates the flipping of a coin. Your program should ask the user how many coins s/he wants to flip and then display the final totals when the simulation is complete. For example, if you type

COIN.FLIP

you should see

HOW MANY?

If you type 10, the program might produce

 # HEADS = 7 # TAILS = 3

Run your program twice, once with an input of 5 and once with an input of 12.

Solution:

```
TO COIN.FLIP
PRINT [HOW MANY?]
FLIP.EM READWORD 0 0
END

TO FLIP.EM :NUMBER :HEADS :TAILS
IF :NUMBER = 0 [( PRINT [# HEADS =]
   :HEADS [# TAILS =] :TAILS ) STOP]
IFELSE ( RANDOM 2 ) = 0
   [MAKE "HEADS :HEADS + 1]
   [MAKE "TAILS :TAILS + 1]
FLIP.EM :NUMBER - 1 :HEADS :TAILS
END
```

Conclusion

The major emphasis of the ICPSC contest is on the local contest site. Each year we get many letters back from local contest directors describing successful contests – some with awards ceremonies and banquets – where the winning team may have solved three problems. (The average number of problems solved in the Junior Logo division this year was two.) If we do our job right, everyone can solve some of the problems and yet, the very best in the world will be challenged to solve all five. The ICPSC provides the problems, sample solutions, instructions for the director and local judges, everything to make it easy for local teachers and administrators to conduct a successful contest. Our goal is simple – to promote the development of computer problem solving skills. Come join us!

1990 Contest

The 10th Annual ICPSC will be held on Saturday, April 28, 1990, with Friday April 27, and Monday, April 30 as alternate contest dates. For more information on how your school or school district can become a contest site, send your request to the address below. You will also receive a free copy of *Compute It!*, the newsletter of the ICPSC.

Direct inquiries to:

Donald T. Piele
ICPSC, P.O. Box 085664

Junior Logo Division Rankings

Rank	Team	School	City, State/Country	Director	Advisor
First	Anne Chazarreta	Bishopstown Community School	Bishopstown, Cork, Ireland	Michael D. Moynihan	
Second	Geraldine Hurley	Bishopstown Community School	Bishopstown, Cork, Ireland	Michael D. Moynihan	
Second	Mark James	Colaiste An Spioraid Naoimh	Bishopstown, Cork, Ireland	Michael D. Moynihan	
Third	Niall Couse	Colaiste An Spioraid Naoimh	Bishopstown, Cork, Ireland	Michael D. Moynihan	
Third	Kyle Rehder Matthew Schniedermann	Ligon Middle School	Raleigh, North Carolina	Cathay Smith	Donna Frink
Fourth	David Frink Charles Powell	Ligon Middle School	Raleigh, North Carolina	Cathay Smith	Donna Frink
Fifth	John Sullivan	Colaiste An Spioraid Naoimh	Bishopstown, Cork, Ireland	Michael D. Moynihan	
Sixth	Amaya Belanger Ines Fdez-Ranada Natalia Martin	Estudio	Madrid, Spain	J. Ernest Berlanger	

Logo: Search and Research

Recurrent Recursion Misconceptions

by Douglas H. Clements

```
TO SHAPE :SIDE
IF :SIDE = 10 STOP
SHAPE :SIDE / 2
REPEAT 4 [FORWARD :SIDE RIGHT 90]
RIGHT 90 FORWARD :SIDE LEFT 90
END
```

```
?SHAPE 80
```

Child 1 (reading IF statement, immediately says): "this makes it draw a square."

Child 2: "[The whole program just] stops, it doesn't draw anything."

Child 3: "If SIDE equals 10 then stop. See, instead of going all forward 80, just go forward 10. Then you're gonna stop. Then you're gonna go. Then (line 3) I guess what you're gonna do is keep on repeating that two times, so it'd be forward about 20 instead of forward 10, forward 20 (line 4), and you're gonna repeat 4, so it'd be forward 80 because it says repeat 4 forward side..." (Kurland & Pea, 1985)

Misconceptions of recursion will come as no surprise to most Logo teachers. Recursion is a difficult concept. Over 50% of sixth to eleventh graders display "buggy" mental models of recursion (Kuspa & Sleeman, 1985). Bugs observed in these students included the following.

1. One student executed program steps in the order in which they appeared. A recursive call was only executed after all other steps were executed. The END statement was ignored.
2. Another considered any use of the parameter as an end condition. This student did not have difficulty passing the parameter. For the following procedures, SHAPE correctly received the parameter 2. The student said, "if shape (which is 2) is greater than 0, then execute the procedure BOX." After executing BOX, however, he ended the program.

```
TO BOX
FORWARD 2 RIGHT 90 FORWARD 2
END
```

```
TO CIRCLE :SHAPE
IF :SHAPE > 0 [BOX]
IF :SHAPE = 0 [STOP]
CIRCLE :SHAPE - 1
END
```

```
?CIRCLE 2
```

3. One did not execute recursive procedures whenever he couldn't determine it's function. That is, if he could not clearly see what would be drawn, he believed program could not be executed ("it doesn't make sense").
4. Several students believed that the value of a parameter is always printed (after which the recursive call was executed by some but not by others).
5. Other students appeared to be "semantically-driven." They insisted that the CIRCLE procedure above would draw a circle.

A study by researchers at Bank Street indicated that only 4 of 15 high school students showed a good understanding of tail recursion. None could follow a program involving embedded recursion (Kurland, Pea, Clement, & Mawby, 1986).

After a year of Logo programming experience, 11- to 12-year-old students were interviewed by these researchers about recursive programs (Kurland & Pea, 1985). The students were asked to think aloud about what small programs would do and to predict via hand-simulation what the turtle would draw. Finally, they were asked to explain any discrepancies between their predictions and an actual run of the program.

There were two related sources of bugs. First, there were general bugs in students' mental models of how Logo code dictates the computer's operations. Some students did only "surface readings" of programs, interpreting each line individually without considering the context provided by previous lines. Other students had the opposite tendency, attributing intentionality to the computer or to the code. Some of these did not differentiate between the meaning of a command line they were simulating from the meaning of lines they expected to follow. For example, one read a line, "If :SIDE equals 100 stop. OK, I think this will make a box that has a hundred side." Such students believed that recursive statements had the *intention* of drawing a square, and thus predicting that the turtle would immediately draw a square before proceeding to the next command. Students also tended to interpret Logo commands in terms of their natural language meanings. For example, they concluded that STOP and END would cause an entire program to halt.

Second, students were fundamentally misled by thinking of recursion as looping, or iteration. For example, they said that a recursive call tells the computer to "go back up," "loop back up," or "go back to the beginning." On the next procedure, that child then predicted that "it's not going to listen to that command...it's going to go past it" and then end.

This mental model persisted even in the face of contradiction between what the program did when run and the students' predictions for what it would do. So, students' errors were systematic. When their mental model failed, they could not rely on the natural language meaning for terms such as STOP. Their tendency to assign intentionality to programs and treat programming more like a human conversation made things worse.

Such misconceptions are also exacerbated by previous programming experience in BASIC. All BASIC children in one study constructed a recursive loop derived from a conceptual model of the GOTO or FOR...NEXT statements (Lee & Lehrer, 1988). None of the others did so.

Children in the BASIC group also did unnecessary nesting. Again, this was the result of attempts to superimpose BASIC's iterative structure onto Logo's recursion. Finally, students had difficulty transferring their knowledge of recursion from turtle graphics to list processing.

In a second study, the researchers tried to address such difficulties instructionally. For example, they introduced recursion with an abstract example of stacking Russian dolls. This was followed by "pseudo-code" descriptions that included a stopping rule and parameter manipulations. With this improved instruction, fewer children displayed misconceptions. However, those with prior BASIC experience were still more likely to employ a looping structure.

Writers have suggested other non-Logo examples of recursion, from recordings to mirrors to drawings to computer simulations (see articles in *Logo Exchange* and *The Computing Teacher* for example; a few are provided in the references). Harvey (1985) provides multiple models of (methods of understanding) recursion and advocates attention to recursive templates such as:

```
TO procedure :INPUT
IF EMPTYP :INPUT [STOP]
do.something.to FIRST :INPUT
procedure BUTFIRST :INPUT
END
```

```
TO procedure :NUMBER
IF EQUALP :NUMBER 0 [STOP]
do.something
procedure :NUMBER - 1
END
```

```
TO procedure :INPUT
IF EMPTYP :INPUT [OUTPUT :INPUT]
OUTPUT combiner (something  FIRST
   :INPUT ) (procedure BUTFIRST
   :INPUT)
END
```

Unfortunately, we still know more about students misconceptions than about how to address them educationally. It is a ripe area for classroom-based research. Next month we will conclude our discussion of misconceptions with a discussion of the "Super Bug."

References

Billstein, R. & Moore, M. L. (1983-84, December/January). Recursion, recursion. *The Computing Teacher*, 11(5), 46-47.

Carter, R. (1983, April). The complete guide to Logo. *Classroom Computer News*, 3(5), 35-39.

Harvey, B. (1985). *Computer science Logo style: Intermediate programming*. Cambridge, MA: MIT Press.

Harvey, G. (1985). *Computer science Logo style: Volume 1*. Cambridge, MA: MIT Press.

Kurland, D. M., & Pea, R. D. (1985). Children's mental models of recursive Logo programs. *Journal of Educational Computing Research*, 235-243, 235-243.

Kurland, D. M., Pea, R. D., Clement, C., & Mawby, R. (1986). A study of the development of programming ability and thinking skills in high school students. *Journal of Educational Computing Research*, 2, 429-458.

Kuspa, L., & Sleeman, D. (1985). *Novice Logo errors*. Unpublished manuscript, Stanford University, Stanford, CA.

Lee, O., & Lehrer, R. (1988). Conjectures concerning the origins of misconceptions in Logo. *Journal of Educational Computing Research*, 4, 87-105.

Riordan, T. (1983-1984, December/January). Helping students with recursion: Teaching strategies. *The Computing Teacher*, 11(5), 38-42.

Riordan, T. (1984, February). Helping students with recursion: Teaching strategies. Part II: Moving to the computer. *The Computing Teacher*, 11(6), 59-64.

Riordan, T. (1984, March). Helping students with recursion: Teaching strategies. Part III: Teaching students about embedded recursion. *The Computing Teacher*, 11(7), 64-69.

Douglas H. Clements
State University of New York at Buffalo
Department of Learning and Instruction
593 Baldy Hall, Buffalo, New York 14260
CIS: 76136,2027; BitNet: INSDHC@UBVMSA